# HAPPY COMPUTER

## TI99/4A *Collection*

## *Year 1983-1984*

# Kleinbuchstaben für den TI 99/4A

**Weil der TI 99/4A keine echten Kleinbuchstaben erzeugt, sondern nur in der Höhe reduzierte Großbuchstaben, stellen wir zwei Programme zur Erzeugung richtiger Kleinschrift vor: ein Basic-Programm und ein Maschinencode-Programm.**

Der einfachste Weg, beim TI 99/4A zu einem Satz echter Kleinbuchstaben mit deutschen Umlauten zu kommen, ist die Umdefinition der Buchstabencharaktere mit dem TI-Basic-Befehl CALL CHAR. Ein Nachteil sei aber nicht verschwiegen: Sobald ein Abbruch des Basic-Programms erfolgt, in dessen Verlauf die Neudefinition stattfand, erscheint wieder der alte Zeichensatz, auch wenn eine Programmfortsetzung mit CONTINUE eingeleitet wird. Trotzdem ist dieser Weg mit Sicherheit der einfachere (Listing 1).

Der Zeichensatz wurde ab ASCII-Wert 97 (Buchstabe a) so umdefiniert, daß er mit dem Pica-Schriftbild bis auf minimale Abweichungen übereinstimmt . Dazu kommen ä, ö, ü und ß, allerdings weicht hier nun die Tastenbelegung vom Original ab. Wenn eine Anpassung an die Belegung eines bestimmten Druckers gewünscht wird, kann man die ASCII-Werte ändern, und zwar in Zeile 170 und 180. Das Programm an sich ist so einfach und durchschaubar, daß eine Beschreibung überflüssig erscheint. Die Strings, welche für die Umdefinition zuständig sind, sind in DATA-Zeilen (Zeilen 130 bis 180) abgespeichert; man könnte sie auch als Datei laden. Die Zeilen 200 bis 240 löschen den Bildschirm und verändern die Farben von Bildschirm (schwarz) und Zeichen (grün). Die Zeilen 250 bis 280 zeigen dem Benutzer Schriftbild und Tastenbelegung, während die Zeilen 290 bis 320 dafür sorgen, daß, wenn eine Taste gedrückt wird, das entsprechende Zeichen auf dem Bildschirm erscheint. Abbrechen kann man das Pro-



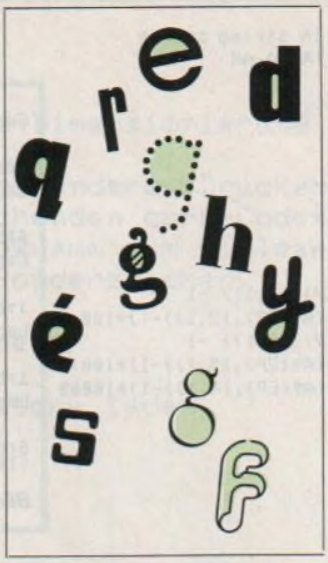
AndreBetz.de 37

```
010 REM Zeichensatz klein
020 REM fuer den TI 99/4(A)
030 REM (C)83 P.Dassow
040 REM Bytes: 1160
050 FOR D=97 TO 114
060 READ C$
070 CALL CHAR(D,C$)
080 NEXT D
090 FOR D=1 TO 8
100 READ N,C$
110 CALL CHAR(N,C$)
120 NEXT D
130 DATA 0000003804 3C443A,00
40404078444438,0000003C40404
03C,0004040 43C44443A,0000003
84478403C
140 DATA 0018242070202020,00
0003C443C0438,C040407844444
444,0010001810101030,0008000
808084830
150 DATA 0040405060504848,00
302020202020 78,0000007854545
454,0000005864444444,0000003
C44444438
160 DATA 0000003844784040,00
0003C38443C0404,0000005824202
020
170 DATA 116,0020207020202018,117,000004444444443A,121,0
000004444 3C0438,64,0018242824242820
180 DATA 129,0044003804 3C443A,130,0044003C44444438,131,0
04400444444443A,128,3C4299A1A199423C
190 REM ENDE DER UMDEF.
200 CALL CLEAR
210 CALL SCREEN(2)
220 FOR ZS=1 TO 13
230 CALL COLOR(ZS,4,2)
240 NEXT ZS
250 PRINT"Ti Kleinbuchstabe
n": :
260 PRINT "ALPHABET:": :"abc
defghijklmnopqrstuvwxyz"
270 PRINT :CHR$(129);" - CTR
L + A": :CHR$(130);" - CTRL
+ B": :CHR$(131);" - CTRL +
C": :
280 PRINT "@ - SHIFT + 2": :
CHR$(128);" - CTRL + ,": : :
 :CHR$(128);"1983 P.Dassow":
 :
290 CALL KEY(5,K,ST)
300 IF ST<>1 THEN 290
310 PRINT CHR$(K);
320 GOTO 290
```

***Listing 1. Basic-Programm zur Erzeugung echter Kleinbuchstaben***

gramm mit FCTN + 4 (CLEAR). Das Programm ist leicht erweiterbar (ändern der Zeile 90) und benötigt lediglich 1,1 KByte. Läßt man die Zeilen 190 bis 320 weg, kann man den eigentlichen Definitionsteil beliebigen Basic-Programmen voranstellen.

(Peter Dassow)



Die Umdefinition mit Hilfe eines Maschinencode-Programms ist dagegen etwas komplizierter, hat aber den Vorteil, daß die Kleinbuchstaben so lange erhalten bleiben, wie der Computer eingeschaltet ist. Außerdem beansprucht das Programm nach Initialisierung nur noch 274 Bytes. Die Hardware-Voraussetzungen sind entweder ein Extended-Basic-Modul oder ein Minimem-Modul, sowie eine RAM-Erweiterung. Um die Sache für solche Leser zugänglich zu machen, die mit Maschinencode nichts im Sinne haben, wurde das Programm in einen Basic-Lader gepackt.

Das Programm in der vorliegenden Form (Listing 2) ist für Extended Basic vorgesehen und wird in den Bereich ab >3000 der RAM-Expansion geladen, von wo es mit CALL LINK (»SMALL«) aufgerufen werden kann. Da das eigentliche Maschinenprogramm nur aus den in Zeile 10040 angegebenen Datas besteht (der Rest sind die neuen Characters), kann es leicht sowohl für andere Speicherbereiche als auch für das Minimem verändert werden. Für das Minimem ergeben sich dann folgende Zeilen:

10040 DATA 2,0,5,240,2,1,114,18,2,2,1,8,4,32,96,40,4,91
10170 FOR ADRESSE = 29184 TO 29457
10210 CALL LOAD (28700,115,18,127,224)
10220 CALL LOAD (32736,83,77,65,76,76,32,114,0)

Selbstverständlich darf im TI-Basic des Minimems das Programm dann nicht mit CALL SMALL angesprochen werden.

(Karl Hagenbuchner)

```
10000 SUB SMALL
10005 !Mag. Karl Hagenbuchner, haidfeldstr. 2,A-4050 Traun
10010 !Kleinbuchstaben und Umlaute
10020 !Pokeliste fuer EXTENDED-BASIC und RAM-Expansion
10030 !Das Programm nimmt 274 Byte ab >3000 in Anspruch
10040 DATA 2,0,5,240,2,1,48,18,2,2,1,8,4,32,32,36,4,91
10050 DATA 0,40,0,56,68,124,68,68,0,40,0,56,68,68,68,56,0,40,0,68,68,68,68,56
10060 DATA 0,0,0,56,8,56,72,52,0,64,64,88,100,68,68,120,0,0,0,60,64,64,64,60
10070 DATA 0,4,4,52,76,68,68,60,0,0,0,56,68,120,64,56,0,24,36,32,112,32,32,32
10080 DATA 0,0,56,68,68,60,4,56,0,64,64,88,100,68,68,68,0,16,0,48,16,16,16,56
10090 DATA 0,8,0,8,8,8,40,16,0,64,64,72,80,96,80,72,0,48,16,16,16,16,16,24
10100 DATA 0,0,0,104,84,84,84,68,0,0,0,88,36,36,36,36,0,0,0,56,68,68,68,56
10110 DATA 0,0,112,72,72,112,64,64,0,0,36,72,72,56,8,8,0,0,0,86,100,64,64,64
10120 DATA 0,0,0,56,64,56,4,120,0,32,32,120,32,32,36,24,0,0,0,72,72,72,72,52
10130 DATA 0,0,0,68,68,40,40,16,0,0,0,68,68,84,84,40,0,0,0,68,40,16,40,68
10140 DATA 0,0,0,136,136,56,8,112,0,0,0,124,8,16,32,124,0,40,0,36,72,72,72,60
10150 DATA 0,40,0,56,72,72,72,56,0,40,0,68,68,68,68,52
10160 CALL INIT
10170 FOR ADRESSE=12288 TO 12561
10180 READ POKE
10190 CALL LOAD(ADRESSE,POKE)
10200 NEXT ADRESSE
10210 CALL LOAD(8194,49,18,63,240)
10220 CALL LOAD(16368,83,77,65,76,76,32,48,0)
10230 CALL LINK("SMALL")
10240 SUBEND
```

***Listing 2. Kleinbuchstaben mit Maschinencode. Geladen wird das Programm trotzdem mit Extended Basic.***



AndreBetz.de 38

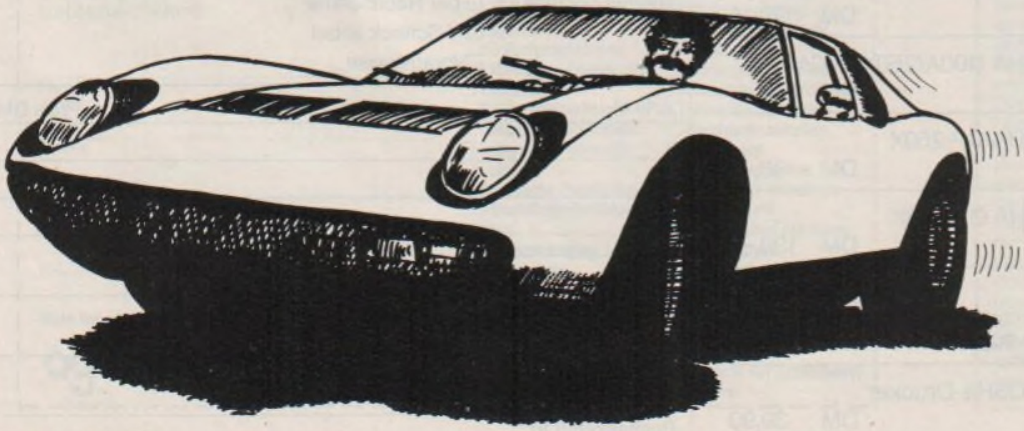
# Spannendes Autorennen mit Hindernissen

**Dieses Spiel verlangt Geschicklichkeit, Reaktionsvermögen und eine gewisse Voraussicht. Beim Straßenrennen steuert der Spieler den Rennwagen mit dem Joystick über einen verschlungenen Pfad mit den verschiedensten Hindernissen, denen es natürlich auszuweichen gilt. Achtlose Zeitgenossen haben Geldscheine verloren, die während der Fahrt aufzusammeln sind.**

Der blaue Rennwagen befindet sich immer am oberen Bildschirmrand, während die Rennstrecke mit den Umleitungen, Felsbrocken, Vorfahrtzeichen und Geldscheinen nach oben wegrollt. Nach dem Starten des Programms wird man zunächst gefragt, ob die Spielregeln benötigt werden. Diese sollten genau studiert werden; besonders das Aussehen der Geldscheine ist von Bedeutung, da im Spiel Hindernisse erscheinen, die eine ähnliche Struktur besitzen, aber einen Zusammenstoß verursachen. Ist dieser Schritt absolviert, wird noch um etwas Geduld gebeten, da die Hindernisstrecke erst noch aufgebaut werden muß. Um das Spiel abwechslungsreicher zu gestalten findet nämlich für jeden Durchgang eine neue zufällige Verstreuung der gegnerischen Rennautos, der Hindernisse und der Banknoten statt — das braucht seine Zeit. Sobald der Summton ertönt, kann es dann aber losgehen. Für Übungszwecke erscheint ein kurzer Streckenabschnitt ohne Hindernisse. Konnte man sich ohne Verluste durchschlängeln, so resultiert die Gesamtpunktzahl aus den Punkten für die zurückgelegte Wegstrecke plus den Zusatzpunkten aus den gefundenen Geldscheinen. Trotzdem sollte man sich jetzt noch nicht auf seinen Lorbeeren ausruhen, denn im zweiten Durchgang wird der horizontale Abstand zwischen den Hindernissen geringer, und die Anforderungen an das Steuergeschick steigen entsprechend. In den Schwierigkeitsstufen drei und vier wird dann der berühmte Überblick verlangt. Hier muß das gute Geld oft »links« liegengelassen werden, um nicht in eine unausweichliche Falle zu tappen. Sollte wider Erwarten der vierte Schwierigkeitsgrad überwunden worden sein, beginnt das ganze Spiel von vorne.

Nachfolgend noch eine kurze Einführung in das Programm. In den Zeilen 110 bis 274 werden die Sonderzeichen definiert. Die Festlegung der Farben findet in den Zeilen 280 bis 440 statt. Bis Zeile 630 liegen die Spielregeln. Die DATA-Zeilen ab 744 legen die Felder neben der Rennbahn, und damit die Form der Rennstrecke, fest. In der Zeile 790 kann man die Länge der Rennstrecke beeinflussen (direkt damit korreliert ist die Wartezeit beim Aufbau der Hindernisse). Die folgenden Zeilen bis einschließlich 1300 bestimmen dann endgültig die zufällige Verteilung der Sonderzeichen. Zeile 1320 zeichnet die Strecke auf dem Bildschirm, und ab 1340 wird der Zustand des Joysticks abgefragt. Zeile 1420 und folgende prüfen, ob ein Geldschein gefunden wurde, ein Zusammenstoß stattfand oder die Strecke verlassen wurde. Zeile 1510 stellt fest, ob eine Runde abgeschlossen werden konnte, wobei Zeile 1580 für die Einstellung der Schwierigkeitsgrade zuständig ist. In Zeile 1640 will das Programm wissen, ob man nach einem Zusammenstoß noch ein Spielchen wagt.

(Brigitte Braun)



***Basicprogramm »Autorennen«***

```
100 RANDOMIZE
110 CALL CHAR(97,"3C2442FFFFFF4242")
120 CALL CHAR(98,"0000000000000C0C")
130 CALL CHAR(99,"3037087C1D3F6660")
135 CALL CHAR(100,"4242422424242424")
140 CALL CHAR(130,"A5FFBD3CBDFFBD18")
145 CALL CHAR(132,"18BDFFBD3CBDFFA5")
150 CALL CHAR(104,"002060F8FC672300")
160 CALL CHAR(105,"400F555500FFFF00")
165 CALL CHAR(110,"007E42040810200 0")
166 CALL CHAR(111,"007E424242427E")
167 CALL CHAR(112,"FF80BBA2BA0ABB80")
168 CALL CHAR(113,"FF01A5BDA5A5A501")
170 CALL CHAR(140,"183C7EFFFF7E3C18")
174 CALL CHAR(141,"0903332707277 7F7")
176 CALL CHAR(142,"0000000242E7FFFF")
178 CALL CHAR(149,"00081031FF311008")
180 CALL CHAR(150,"00000040FF4")
190 CALL CHAR(151,"00000002FF02")
200 CALL CHAR(152,"3C42818DFD81423C")
210 CALL CHAR(153,"9999999999999999")
220 CALL CHAR(81,"002F69A929292F")
230 CALL CHAR(155,"03070E1C3870E0C")
240 CALL CHAR(91,"FF80BB8ABAA2BB80")
250 CALL CHAR(92,"FF01A5BDA5A5A501")
260 CALL CHAR(156,"00000B0B0B0B")
270 CALL CHAR(120,"FFFFFFFFFFFFFFFF")
272 CALL CHAR(121,"010303070F1F3F7F")
274 CALL CHAR(122,"7F3F1F0F070301")
280 B=1
290 CALL COLOR(12,15,1)
300 C=1
310 FOR I=2 TO 6
320 B=B+1
```



AndreBetz.de 52

```
330 C=C+2
340 CALL COLOR(8,C,16)
350 NEXT I
360 CALL COLOR(13,5,15)
370 CALL COLOR(14,2,15)
380 CALL COLOR(15,14,15)
390 CALL COLOR(16,10,16)
400 CALL COLOR(10,16,5)
405 CALL COLOR(11,7,16)
410 CALL COLOR(9,2,16)
420 CALL COLOR(8,4,16)
430 CALL COLOR(7,6,16)
440 CALL CLEAR
450 PRINT "++++++++++++++++":"+STRASSENRENNEN+":"++++++++++++++++":::::::::::::::
:::
460 PRINT "BENOETIGEN SIE DIE":"SPIELREGELN? /J/ /N/"
470 CALL KEY(0,K,S)
480 IF S=0 THEN 470
490 IF K=74 THEN 530
500 IF K=78 THEN 640
510 CALL SOUND(200,-1,0)
520 GOTO 470
530 CALL CLEAR
540 PRINT "SIE SOLLEN EINEN WAGEN,":"MIT ";CHR$(130);" ANGEDEUTET,"
550 PRINT "DURCH EINEN HINDERNISKURS":"MANOEVRIEREN! MITTELS DES "
560 PRINT "JOYSTICKS BEWEGEGEN SIE ":"DEN WAGEN NACH LINKS ODER";" RECHTS."
570 PRINT "DIE GELDSCHEINE AUF DER":"STRASSE SOLLTEN SIE NICHT":"LIEGENLASSEN, S
IE BEDEUTEN":"PLUSPUNKTE!"
580 PRINT "DAS SPIEL BEGINNT BEI DER ":"SCHWIERIGKEITSTUFE 1 UND":"ENDET MIT DER
 STUFE 4."
590 PRINT "SOLLTEN SIE ALLES HEIL UEBER":"STANDEN HABEN, SO BE":"KOMMEN SIE EIN
FREISPIEL"::
600 PRINT TAB(7);CHR$(81);"  =  10":TAB(7);CHR$(91)&CHR$(92);"  =  20":TAB(7);CHR
$(156);"  = 100"
605 PRINT TAB(7);CHR$(112)&CHR$(113);"  =  50"
610 PRINT "ALLES KLAR?":
620 INPUT "J ODER N EINGEBEN:":X$
630 IF X$()"J" THEN 540
640 A=17
650 P=0
660 S=5
670 DIM S$(200)
680 CALL CLEAR
690 PRINT TAB(5);"BITTE WARTEN"::::::::
700 PRINT "JETZTIGE PUNKTZAHL: ";P::::
710 PRINT "BISHERIGE ":
720 PRINT "HOECHSTPUNKTZAHL:  =";H::::
730 PRINT "ABSOLUTE ":
740 PRINT "HOECHSTPUNKTZAHL:   ";5276:::
744 DATA "
745 DATA "
746 DATA "
747 DATA "
748 DATA "
749 DATA "
750 DATA "
751 DATA "
752 DATA "
753 DATA "
754 DATA "
755 DATA "
756 DATA "
757 DATA "
758 DATA "
760 B$=CHR$(120)&CHR$(120)&CHR$(120)&CHR$(120)&CHR$(120)
765 A=15
770 E$=CHR$(120)&CHR$(91)&CHR$(92)&CHR$(120)&CHR$(120)
780 G$=CHR$(120)&CHR$(120)&CHR$(120)&CHR$(81)&CHR$(120)
785 H$=CHR$(120)&CHR$(120)&CHR$(120)&CHR$(112)&CHR$(113)
787 RESTORE 744
790 FOR I=1 TO 150
791 IF I/43=INT(I/43)THEN 794
792 READ D$,A$
793 GOTO 800
794 RESTORE 744
800 IF I/S()INT(I/S)THEN 880
810 X=INT(9*RND)+1
820 ON (INT(3*RND)+1)GOTO 830,850,855
830 ON X GOSUB 900,940,980,1020,1060,1100,1140,1180,1220
840 GOTO 860
850 ON X GOSUB 920,960,1000,1040,1080,1120,1160,1200,1240
853 GOTO 860
855 ON X GOSUB 925,965,1005,1045,1085,1125,1165,1205,1245
860 NEXT I
870 GOTO 1260
880 S$(I)=D$&B$&B$&A$
890 GOTO 860
900 S$(I)=D$&CHR$(140)&CHR$(97)&CHR$(104)&CHR$(140)&CHR$(120)&G$&A$
910 RETURN
920 S$(I)=D$&CHR$(97)&CHR$(97)&CHR$(97)&CHR$(120)&CHR$(120)&E$&A$
923 RETURN
925 S$(I)=D$&CHR$(120)&CHR$(141)&CHR$(141)&CHR$(120)&CHR$(142)&H$&A$
927 RETURN
930 RETURN
940 S$(I)=D$&CHR$(151)&CHR$(153)&CHR$(150)&CHR$(152)&CHR$(81)&B$&A$
950 RETURN
960 S$(I)=D$&E$&CHR$(120)&CHR$(98)&CHR$(99)&CHR$(140)&CHR$(120)&A$
963 RETURN
965 S$(I)=D$&CHR$(110)&CHR$(111)&CHR$(120)&CHR$(152)&CHR$(120)&B$&A$
970 RETURN
980 S$(I)=D$&G$&CHR$(120)&CHR$(120)&CHR$(140)&CHR$(120)&CHR$(140)&A$
990 RETURN
1000 S$(I)=D$&CHR$(153)&CHR$(152)&CHR$(120)&CHR$(104)&CHR$(105)&G$&A$
1003 RETURN
1005 S$(I)=D$&CHR$(142)&CHR$(120)&CHR$(132)&CHR$(132)&CHR$(120)&B$&A$
1010 RETURN
1020 S$(I)=D$&CHR$(151)&CHR$(153)&CHR$(150)&CHR$(120)&CHR$(120)&E$&A$
1030 RETURN
1040 S$(I)=D$&CHR$(151)&CHR$(156)&CHR$(150)&CHR$(152)&CHR$(91)&B$&A$
1043 RETURN
1045 S$(I)=D$&CHR$(120)&CHR$(120)&CHR$(120)&CHR$(149)&CHR$(149)&B$&A$
1050 RETURN
1060 S$(I)=D$&E$&CHR$(120)&CHR$(155)&CHR$(151)&CHR$(156)&CHR$(150)&A$
1070 RETURN
1080 S$(I)=D$&G$&CHR$(153)&CHR$(153)&CHR$(153)&CHR$(153)&CHR$(120)&A$
1083 RETURN
1085 S$(I)=D$&CHR$(112)&CHR$(113)&CHR$(120)&CHR$(100)&CHR$(100)&B$&A$
1090 RETURN
1100 S$(I)=D$&CHR$(155)&CHR$(155)&CHR$(155)&CHR$(155)&CHR$(155)&B$&A$
1110 RETURN
1120 S$(I)=D$&B$&CHR$(97)&CHR$(120)&CHR$(120)&CHR$(97)&CHR$(99)&A$
1123 RETURN
1125 S$(I)=D$&H$&CHR$(110)&CHR$(111)&CHR$(120)&CHR$(142)&CHR$(142)&A$
1130 RETURN
1140 S$(I)=D$&B$&CHR$(155)&CHR$(155)&CHR$(155)&CHR$(155)&CHR$(155)&A$
1150 RETURN
1160 S$(I)=D$&CHR$(152)&CHR$(153)&CHR$(152)&CHR$(120)&CHR$(152)&CHR$(120)&CHR$(9
9)&CHR$(120)&CHR$(99)&CHR$(81)&A$
1163 RETURN
1165 S$(I)=D$&B$&CHR$(99)&CHR$(141)&CHR$(120)&CHR$(120)&CHR$(149)&A$
1170 RETURN
1180 S$(I)=D$&B$&"STOP"&CHR$(120)&A$
1190 RETURN
1200 S$(I)=D$&CHR$(120)&CHR$(120)&CHR$(98)&CHR$(99)&CHR$(120)&B$&A$
1203 RETURN
1205 S$(I)=D$&B$&CHR$(112)&CHR$(113)&CHR$(132)&CHR$(120)&CHR$(132)&A$
1210 RETURN
1220 S$(I)=D$&CHR$(99)&"STOP"&B$&A$
1230 RETURN
1240 S$(I)=D$&CHR$(81)&CHR$(152)&CHR$(97)&CHR$(97)&CHR$(152)&CHR$(81)&CHR$(97)&C
HR$(99)&CHR$(120)&CHR$(152)&A$
1243 RETURN
1245 S$(I)=D$&B$&CHR$(132)&CHR$(120)&CHR$(120)&CHR$(110)&CHR$(111)&A$
1250 RETURN
1260 CALL SOUND(500,-3,5)
1265 A$=""
1270 C$=A$&B$&B$&A$
1280 FOR I=1 TO 24
1290 PRINT C$
1300 NEXT I
1310 Z=0
1320 CALL HCHAR(1,A,130)
1330 CALL SOUND(-500,120,5,-7,7)
1340 CALL JOYST(1,J1,J2)
1350 IF J1-J2+J1=0 THEN 1420
1360 IF J1=-4 THEN 1380
1370 IF J1=4 THEN 1400
1380 A=A-1
1390 GOTO 1420
1400 A=A+1
1410 GOTO 1420
1420 CALL GCHAR(2,A,PIC)
1430 IF PIC=91 THEN 1840
1440 IF PIC=92 THEN 1840
1450 IF PIC=81 THEN 1870
1455 IF PIC=112 THEN 1930
1456 IF PIC=113 THEN 1930
1460 IF PIC=156 THEN 1900
1470 IF PIC()120 THEN 1520
1480 P=P+1
1490 Z=Z+1
1500 PRINT S$(Z)
1510 IF Z=172 THEN 1710 ELSE 1320
1520 CALL SOUND(200,-6,0)
1530 CALL CLEAR
1540 PRINT TAB(5);"ZUSAMMENSTOSS!"::::
1550 PRINT "PUNKTZAHL:";P
1560 IF P(=H THEN 1630
1570 H=P
1580 FOR SI=1 TO 5
1590 FOR SJ=-3 TO -1
1600 CALL SOUND(150,SJ,3)
1610 NEXT SJ
1620 NEXT SI
1630 PRINT "HIGHSCORE:";H::::::
1640 PRINT "NOCH EIN SPIEL? /J/ /N/"
1650 CALL KEY(0,K,S)
1660 IF S=0 THEN 1650
1670 IF K=74 THEN 640
1680 IF K=78 THEN 1820
1690 CALL SOUND(200,-1,0)
1700 GOTO 1650
1710 CALL CLEAR
1720 PRINT TAB(4);"GRATULATION"::::
1730 PRINT "SIE HABEN DIE STRECKE MIT"::"DEM SCHWIERIGKEITSGRAD";"";ABS(S-6);"":
:
1740 PRINT "BEWAELTIGT."::::::::
1750 S=S-1
1760 IF S)1 THEN 1780
1770 S=4
1780 PRINT "ES GEHT WEITER MIT":::"STUFE";ABS(S-6)::::
1790 FOR V=1 TO 1500
1800 NEXT V
1810 GOTO 680
1820 CALL SOUND(500,880,1)
1830 END
1840 CALL SOUND(-100,660,5)
1850 P=P+20
1860 GOTO 1490
1870 CALL SOUND(-500,444,5)
1880 P=P+10
1890 GOTO 1490
1900 CALL SOUND(-500,1760,1)
1910 P=P+100
1920 GOTO 1490
1930 CALL SOUND(-500,1000,3)
1940 P=P+50
1950 GOTO 1490
```

***Basicprogramm »Autorennen« (Schluß)***



AndreBetz.de 53

nen Basic-Befehle schnell den gewünschten Effekt für Ihr eigenes Programm »zurecht schneidern« können. SAVEn Sie das Programm am besten nach der Eingabe mit GOTO 3. Es wird sich dann beim nächsten Laden selbst starten.

Einzeln laufen die Teilprogramme auch auf dem ZX81 mit 1KByte RAM. Wegen der geringen Speicher-Kapazität sind dann natürlich bei den Teilprogrammen, die den ganzen Bildschirm ausfüllen, kürzere Strings zu verwenden, wie das Beispiel in Bild 2 zeigt, was aber die grafische Wirkung nicht beeinträchtigt.

(Gerd Müssig)

```
850 NEXT X
900 REM WIEDERHOLUNG
910 PRINT AT 10,0;"                "
915 PRINT AT 9,0;"                "
917 PRINT AT 11,0;"                "
920 GOTO 5
```

*Bild 1. Einige Programmroutinen zur Erzeugung optischer Bonbons als Kosmetik für müde Programme (Fortsetzung)*

# Damit im Haushalt die Kasse stimmt

**Das Programm »Haushaltskasse« für den TI 99/4A dient zur Eingabe und Speicherung der Ausgaben eines privaten Haushalts für jeweils einen Monat. Die Ausgaben können dann entweder auf dem Bildschirm durch Säulengrafiken dargestellt oder nach bestimmten Kriterien, wie Tag oder Artikelgruppe, aufgelistet werden.**

```
10 ! HAUSHALTSKASSE
20 ! FUER TI99/4A EXTENDED   BASIC ,VERSION 110
30 ! COPYRIGHT 1983 BY   HANS-JOACHIM BIGALKE
40 ! SEUFFERTSTR.1
50 ! 87 WUERZBURG 1
60 CALL CLEAR
70 VF=2 :: HF=8 :: BF=7 :: GOSUB 2480 !DEF.DER FARBEN(KOENNEN GEAENDERT WERDEN)
80 DISPLAY AT(11,1):"**************************":"H A U S H A L T S K A S S E":
"**************************"
90 CALL KEY(0,K,S):: IF S=0 THEN 90
100 DIM S$(102,5),M(31),D(10),C$(10),B$(10)
110 C$(1)="LEBENSMITTEL" :: C$(2)="HAUSHALTSWAREN" :: C$(3)="WERKZEUG" :: C$(4)=
"KLEIDUNG" :: C$(5)="SCHREIBWAREN" :: C$(6)="VERGNUEGUNGEN"
120 C$(7)="VERSCHIEDENES" :: C$(8)="UNNOETIGES" :: C$(0)="GESAMTES"
130 !ARTIKELGRUPPEN KOENNEN GEANDERT WERDEN
140 B$(1)="DATUM        :" :: B$(2)="ART.GRUPPE   :" :: B$(3)="ARTIKEL(?=UNNOETI
G):" :: B$(4)="WO GEKAUFT   :" :: B$(5)="PREIS        :"
150 Z1=0 :: Z=0
160 CALL CLEAR
170 GOTO 180 :: CALL SPRITE :: CALL POSITION :: CALL LOCATE :: CALL DELSPRITE ::
 CALL SCREEN :: CALL COLOR :: CALL SOUND :: CALL KEY :: CALL CHAR :: CALL HCHAR
:: CALL VCHAR :: CALL CLEAR :: VF :: BF :: HF
180 GOTO 190 :: SUM :: R1 :: X :: XS :: YS :: Z1 :: Z :: K :: M1$ :: A :: T :: S
 :: TA$ :: AN$ :: I :: NV$ :: N$ :: R :: G :: NAX :: GES :: TAG :: Y :: F :: SOR
T :: L :: J$ :: TA :: TB :: DES :: DAX :: H :: HO :: SP :: SI :: !@P-
190 ! HAUPTMENUE
200 CALL CLEAR
210 VF=2 :: HF=7 :: BF=6 :: GOSUB 2480
220 DISPLAY AT(12,8):"1-DATEN EINLESEN":"       2-MENUE":"       3-ENDE"
230 CALL KEY(0,K,S):: IF S=0 THEN 230
240 IF K)51 OR K(49 THEN 230
250 ON K-48 GOTO 260,280,2650
260 GOSUB 930
270 GOTO 190
280 ! MENUE
290 CALL CLEAR
300 VF=2 :: HF=6 :: BF=5 :: GOSUB 2480
310 DISPLAY AT(12,8):"1=EINGABE":"      2=GRAFIK ":"      3=AUSGABE ":"
4=SORTIEREN"
320 DISPLAY AT(16,8):"5=SPEICHERN ":"      6=RUECKSPRUNG"
330 CALL KEY(0,K,S):: IF S=0 THEN 330
340 IF K)54 OR K(49 THEN 330
350 ON K-48 GOSUB 390,1170,650,2490,1060,380
360 IF K-48=6 THEN 190
370 GOTO 280
380 RETURN
390 ! EINGABE
400 CALL CLEAR
410 VF=2 :: HF=4 :: BF=4 :: GOSUB 2480
420 INPUT "MONAT  ?":M1$
430 CALL CLEAR
440 Z1=Z1+1
450 Z=Z1
460 CALL CLEAR :: FOR A=1 TO 7
470 DISPLAY AT(A,1):A;"=";C$(A)
480 NEXT A
490 FOR K=1 TO 5
```

*Listing »Haushaltskasse«*



AndreBetz.de 92

```
500 IF K=3 OR K=4 THEN T=1 :: S=14 ELSE T=0 :: S=0
510 DISPLAY AT(2*K+10,1):B$(K):: DISPLAY AT(2*K+10+T,15-S):S$(Z,K)
520 NEXT K
530 DISPLAY AT(10,1):Z :: DISPLAY AT(12,17):".";M1$
540 FOR K=1 TO 5
550 IF K=1 THEN ACCEPT AT(2*K+10,15)BEEP VALIDATE(NUMERIC)SIZE(-2):S$(Z,K)ELSE I
F K=3 OR K=4 THEN ACCEPT AT(2*K+11,1)SIZE(-32):S$(Z,K)
560 IF K=2 THEN ACCEPT AT(2*K+10,15)BEEP VALIDATE("1234567")SIZE(-1):S$(Z,K)
570 IF K=5 THEN ACCEPT AT(2*K+10,15)BEEP VALIDATE("0123456789.")SIZE(-8):S$(Z,K)
580 IF (K=1 OR K=2 OR K=5)AND S$(Z,K)="" THEN 550
590 IF K=1 AND VAL(S$(Z,1))>31 THEN 550
600 NEXT K
610 DISPLAY AT(23,1):"NEUE EINGABE J/N K=KORREKTUR"
620 CALL KEY(0,K,S):: IF S=0 THEN 620
630 IF K=74 THEN 430 ELSE IF K=75 THEN 490 ELSE 640
640 RETURN
650 ! AUSGABE MONAT
660 CALL CLEAR
670 VF=5 :: HF=16 :: BF=10 :: GOSUB 2480
680 DISPLAY AT(1,1):"KORREKTUR=K";" ENDE=E"
690 DISPLAY AT(2,1):"MONAT ";M1$
700 DISPLAY AT(4,1):"1=GESAMT":"2=EINZELNER TAG":"3=EINZELNE ARTIKELGRUPPEN"
710 CALL KEY(0,R,S):: IF S=0 THEN 710 ELSE IF R<49 OR R>51 THEN 710 ELSE R=R-48
720 IF R=1 THEN 740 ELSE IF R=2 THEN DISPLAY AT(9,1):"TAG?" :: ACCEPT AT(9,6)VAL
IDATE("1234567890")SIZE(2)BEEP:TA$
730 IF R=3 THEN DISPLAY AT(9,1):"ART.NR?" :: FOR I=1 TO 7 :: DISPLAY AT(15+I,4):
I;C$(I):: NEXT I :: ACCEPT AT(9,10)VALIDATE("1234567")SIZE(1)BEEP:AN$
740 CALL CLEAR :: FOR I=1 TO Z1
750 IF R=0 OR R1=0 THEN 770
760 IF R=2 AND R1=3 AND S$(I,1)=TA$ AND S$(I,2)=AN$ THEN 780 ELSE 880
770 IF R=1 THEN 780 ELSE IF R=2 AND S$(I,1)=TA$ THEN 780 ELSE IF R=3 AND S$(I,2)
=AN$ THEN 780 ELSE 880
780 FOR K=1 TO 5
790 PRINT B$(K);S$(I,K): :
800 NEXT K
810 SUM=SUM+VAL(S$(I,5)):: PRINT : : : : : :
820 CALL KEY(0,K,S):: IF S=0 THEN 820
830 IF K=75 THEN 860
840 IF K=69 THEN 910
850 GOTO 880
860 Z=I :: GOSUB 460
870 CALL CLEAR
880 NEXT I
890 CALL CLEAR :: DISPLAY AT(15,1):"SUMME =";SUM;"DM"
900 CALL KEY(0,K,S):: IF S=0 THEN 900
910 SUM,R,R1=0
920 RETURN
930 ! DATEN MONAT EINLESEN
940 CALL CLEAR
950 INPUT "GEBEN SIE BITTE DAS CODEWORT EIN! ":NV$
960 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 192
970 INPUT #1:Z1,M1$,N$
980 IF N$<>NV$ THEN 1030
990 FOR I=1 TO Z1 STEP 2
1000 INPUT #1:S$(I,1),S$(I,2),S$(I,3),S$(I,4),S$(I,5),S$(I+1,1),S$(I+1,2),S$(I+1
,3),S$(I+1,4),S$(I+1,5)
1010 NEXT I
1020 GOTO 1040
1030 CALL CLEAR :: CALL SOUND(4000,-3,0):: DISPLAY AT(12,1):"FALSCHES CODEWORT!"
1040 CLOSE #1
1050 RETURN
1060 ! DATEN  SPEICHERN
1070 CALL CLEAR
1080 VF=16 :: HF=6 :: BF=7 :: GOSUB 2480
1090 INPUT "GEBEN SIE BITTE DAS CODEWORT EIN!":N$
1100 OPEN #3:"CS1",INTERNAL,OUTPUT,FIXED 192
1110 PRINT #3:Z1,M1$,N$
1120 FOR I=1 TO Z1 STEP 2
1130 PRINT #3:S$(I,1),S$(I,2),S$(I,3),S$(I,4),S$(I,5),S$(I+1,1),S$(I+1,2),S$(I+1
,3),S$(I+1,4),S$(I+1,5)
1140 NEXT I
1150 CLOSE #3
1160 RETURN
1170 ! GRAFIK
1180 CALL CLEAR
1190 VF=2 :: HF=4 :: BF=12 :: GOSUB 2480
1200 DISPLAY AT(12,4):" 1=GESAMTER MONAT ";M1$:"    2=EINZELNE TAGE"
1210 CALL KEY(0,R,S):: IF S=0 THEN 1210
1220 IF R>50 OR R<49 THEN 1210
1230 G=R-48
1240 IF G=2 THEN 1870
```

*Listing »Haushaltskasse« (Fortsetzung)*

# Haushaltskasse

Manche Mark könnte man im Haushalt einsparen, wenn man eine Übersicht über die Arten und Mengen der Ausgaben hätte. Dadurch wird zwar nichts billiger, aber vielleicht das eine oder andere als nicht unbedingt nötig erkannt. Die Anlage und Führung eines Haushaltsbuches — wie es unsere Mütter noch im Haushaltskundeunterricht lernen mußten — ist zeitraubend und umständlich. Überlassen wir das ruhig unserem elektronischen Helfer, dem Computer.

Das Programm »Haushaltskasse« nützt die Möglichkeiten des TI 99/4A weidlich aus. Es nimmt nicht nur die Rechenarbeit ab, sondern stellt die Ergebnisse auch grafisch anschaulich dar. Zur Programmeingabe in den Rechner sind allerdings einige Tricks nötig. Eine Reihe von Zeilen sind so lang, daß man sie nicht in einem Stück eintippen kann. Man beendet dann eine solche Eingabe vorläufig, holt die Programmzeile erneut zurück und kann den Rest ergänzen. Die Zeile 180 (siehe Listing) stellt einen anderen Trick dar.

Obwohl der Befehl »GOTO 190« den Interpreter veranlaßt, den Rest der Zeile 180 zu überspringen, bewirkt dieser Rest, daß die dort aufgeführten Variablen nicht auf den Speicherplatz hin untersucht werden. Diese Maßnahme beschleunigt den Programmablauf. Angewendet werden kann dieser Teil aber nur in der Extended-Basic-Version 110. Soll das Programm mit der Version 100 laufen, muß die Zeile 180 wegfallen. Die Bezeichnungen der Artikelgruppen können den eigenen Bedürfnissen angepaßt werden, indem man die neuen Bezeichnungen in die Zeilen 110 und 120 einschreibt.

Beim Programmstart erscheint der Programmname »Haushaltskasse« auf dem Bildschirm: Nach Drücken einer beliebigen Taste wird das Grundmenü aufgelistet und man hat die Wahl zwischen:

1. Daten einlesen (der be-



AndreBetz.de 93

```
1250 !GRAFIK GESAMTER MONAT
1260 CALL CLEAR
1270 VF=7 :: HF=12 :: BF=2 :: GOSUB 2480
1280 DISPLAY AT(12,4):" 1=GESAMT":"    2=ARTIKELGRUPPE":"    3=UNNOETIGE AUSGABE
N":"    4=RUECKSPRUNG"
1290 CALL KEY(0,K,S):: IF S=0 THEN 1290
1300 IF K=49 THEN 1380 ELSE IF K=50 THEN 1310 ELSE IF K=51 THEN 1360 ELSE IF K=5
2 THEN 1860 ELSE 1290
1310 CALL CLEAR :: FOR A=1 TO 7
1320 DISPLAY AT(12+A,2):A;"=";C$(A)
1330 NEXT A
1340 DISPLAY AT(21,2):"ARTIKELNUMMER?" :: ACCEPT AT(21,25)SIZE(-1)VALIDATE("1234
567")BEEP:A
1350 GOTO 1390
1360 A=8
1370 GOTO 1390
1380 A=0
1390 MAX=0
1400 GES=0
1410 TAG=0
1420 FOR I=1 TO 31
1430 M(I)=0
1440 NEXT I
1450 CALL CLEAR
1460 VF=2 :: HF=14 :: BF=14 :: GOSUB 2480
1470 REM
1480 FOR I=1 TO Z1
1490 IF A=0 THEN 1530 ELSE IF A=8 THEN 1510 ELSE 1500
1500 IF VAL(S$(I,2))=A THEN 1530 ELSE 1540
1510 IF S$(I,3)="" THEN 1540
1520 IF SEG$((S$(I,3)),LEN(S$(I,3)),1)="?" THEN 1530 ELSE 1540
1530 M(VAL(S$(I,1)))=M(VAL(S$(I,1)))+VAL(S$(I,5))
1540 NEXT I
1550 FOR I=1 TO 31
1560 IF M(I))MAX THEN 1570 ELSE 1590
1570 MAX=M(I)
1580 TAG=I
1590 GES=GES+M(I)
1600 NEXT I
1610 DISPLAY AT(4,1):"SIE HABEN IM MONAT ";M1$;" FUER ";C$(A);" DM";GES;"AUSGEGE
BEN.AM";TAG;" WAREN DIE AUSGABEN MIT";MAX;" DM AM GROESSTEN"
1620 DISPLAY AT(23,1):"BELIEBIGE TASTE DRUECKEN"
1630 ! GRAFIK AUF BILDSCHIRM
1640 CALL KEY(0,K,S):: IF S=0 THEN 1640
1650 CALL CLEAR
1660 VF=2 :: HF=4 :: BF=13 :: GOSUB 2480
1670 Y=MAX/17
1680 DISPLAY AT(1,1):C$(A);" ";M1$
1690 DISPLAY AT(23,1):"2345671234567123456712345671"
1700 CALL HCHAR(23,31,50)
1710 CALL HCHAR(23,32,51)
1720 DISPLAY AT(24,4):"1 KAESTCHEN=DM";INT(Y*100*17/18+0.5)/100
1730 FOR I=1 TO 31
1740 IF M(I)=0 THEN 1760
1750 CALL VCHAR(21-INT(M(I)/Y),I+1,30,INT(M(I)/Y)+1)
1760 NEXT I
1770 R,R1,X,XS,YS=0 :: TA$,AN$=""
1780 CALL KEY(3,K,S):: IF S=0 THEN 1780
1790 IF K=32 THEN 1800 ELSE 1860
1800 CALL SPRITE(#1,65,2,168,8)
1810 CALL KEY(0,K,S):: IF K=83 THEN X=-8 ELSE IF K=68 THEN X=8 ELSE IF K=49 THEN
 1840 ELSE 1810
1820 CALL POSITION(#1,YS,XS):: IF XS<16 THEN XS=16 ELSE IF XS)240 THEN XS=240
1830 CALL LOCATE(#1,YS,XS+X):: GOTO 1810
1840 CALL POSITION(#1,YS,XS):: XS=INT(XS/8):: TA$=STR$(XS):: R=2 :: IF A)0 AND A
<8 THEN AN$=STR$(A):: R1=3
1850 CALL DELSPRITE(#1):: GOSUB 740
1860 RETURN
1870 ! GRAFIK EINZELNER TAG
1880 CALL CLEAR
1890 VF=2 :: HF=3 :: BF=6 :: GOSUB 2480
1900 FOR I=1 TO 7
1910 D(I)=0
1920 NEXT I
1930 DISPLAY AT(12,4):"MONAT ";M1$
1940 DISPLAY AT(15,1):"TAGE?     VOM    . BIS    ." :: ACCEPT AT(15,15)BEEP SIZE(-
2)VALIDATE("1234567890"):TA :: ACCEPT AT(15,23)BEEP SIZE(-2)VALIDATE("1234567890
"):TB
1950 DES=0
1960 FOR I=1 TO Z1
1970 IF VAL(S$(I,1))=TA AND VAL(S$(I,1))<=TB THEN 1980 ELSE 2000
```

*Listing »Haushaltskasse« (Fortsetzung)*

reits auf Kassette gespeicherten Daten),
2. Menü (Hauptmenü) und
3. Ende.

Vor dem Einlesen eventuell schon vorhandener Daten vom Band muß übrigens noch ein Codewort eingetippt werden, das beim Abspeichern vorgegeben wurde — Datenschutz im Haushalt.

Mit Wahl der Ziffer 2 erscheint das Hauptmenü auf dem Bildschirm:
1. Eingabe (neuer Daten),
2. Grafik (bildliche Darstellung oder Rechenergebnisse),
3. Ausgabe (numerisch nach Tagen oder Artikeltypen),
4. Sortieren (nach Tagen oder Artikeltypen),
5. Speichern (auf Kassette) und
6. Rücksprung (zum Grundmenü).

Nach der Wahl der Ziffer 1 muß der Anwender als erstes den Monat eingeben, dem die Daten zugeordnet werden sollen. Anschließend erscheint eine Eingabemaske für
- ☐ das Datum,
- ☐ die Artikelgruppe,
- ☐ die Frage, ob die Anschaffung als unnötig oder notwendig erachtet wird,
- ☐ die Frage, »wo gekauft« wurde und
- ☐ den Preis.

Eingaben, die offensichtlich falsch sind, werden zurückgewiesen. Die Frage, ob ein Artikel unnötig war, kann auch durch ein Fragezeichen beantwortet werden. In der späteren Auswertung erscheint der Posten dann unter den unnötigen Ausgaben. Sind alle Daten eingegeben worden, fragt der Computer, ob noch eine Eingabe oder Korrektur gewünscht wird.

Unter dem Menüpunkt 2 erfolgt eine grafische und numerische Auswertung der Daten. Wieder erscheint ein kleines Menü mit den Wahlmöglichkeiten:
1. Gesamter Monat,
2. Einzelne Tage,

Wählt man die »1«, leuchtet das nächste Menü vom Bildschirm. Es zeigt:
1. Gesamt,
2. Artikelgruppen,
3. Unnötige Ausgaben,
4. Rücksprung.



AndreBetz.de 94

```
1980 D(VAL(S$(I,2)))=D(VAL(S$(I,2)))+VAL(S$(I,5))
1990 DES=DES+VAL(S$(I,5))
2000 NEXT I
2010 CALL CLEAR
2020 DAX=0
2030 DISPLAY AT(1,1):"SIE HABEN IN DER ZEIT VOM";TA;".";"BIS";TB;".";M1$;"FOLGEN
DES AUSGEGEBEN:"
2040 FOR I=1 TO 7
2050 DISPLAY AT(2*I+5,1):C$(I):: DISPLAY AT(2*I+5,17):USING "#######.##":D(I):: D
ISPLAY AT(2*I+5,27):"DM"
2060 IF D(I)<=DAX THEN 2080
2070 DAX=D(I)
2080 NEXT I
2090 DISPLAY AT(22,1):"SUMME=DM " :: DISPLAY AT(22,17):USING "#######.##":DES
2100 Y=DAX/18
2110 CALL KEY(0,K,S)
2120 IF S=0 THEN 2110
2130 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
2140 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
2150 CALL CHAR(105,"0103070F1F3F7FFF")
2160 CALL CHAR(112,"0103070F1F3F7FFF")
2170 CALL CHAR(121,"0103070F1F3F7FFF")
2180 CALL COLOR(9,4,1)
2190 CALL COLOR(10,5,8)
2200 CALL COLOR(11,3,5)
2210 CALL COLOR(12,8,3)
2220 CALL CLEAR
2230 CALL SCREEN(15)
2240 DISPLAY AT(1,1):"ZEIT VOM";TA;". BIS";TB;".";M1$
2250 I=1
2260 FOR H=49 TO 57
2270 IF H>55 THEN 2290
2280 CALL HCHAR(22,I*2+4,H,1)
2290 CALL VCHAR(22-I*2,20,H,1)
2300 I=I+1
2310 NEXT H
2320 DISPLAY AT(2,18):"10=";DAX :: DISPLAY AT(3,22):"DM"
2330 FOR I=1 TO 7
2340 IF D(I)=0 THEN 2440
2350 D(I)=INT(D(I)/Y)
2360 HO=22-D(I)
2370 SP=I*2+4
2380 SI=HO-1
2390 CALL VCHAR(HO,SP,96,D(I))
2400 CALL VCHAR(SI,SP,121,1)
2410 CALL VCHAR(HO,SP+1,104,D(I))
2420 CALL VCHAR(SI,SP+1,105,1)
2430 CALL VCHAR(21,SP+1,112,1)
2440 NEXT I
2450 FOR I=5 TO 8 :: CALL COLOR(I,2,3):: NEXT I :: FOR I=1 TO 7 :: FOR L=LEN(C$(
I))TO 1 STEP -1 :: CALL VCHAR(20-LEN(C$(I))+L,2*I+4,ASC(SEG$(C$(I),L,1)),1):: NE
XT L :: NEXT I
2460 CALL KEY(0,K,S):: IF S=0 THEN 2460
2470 RETURN
2480 FOR I=1 TO 14 :: CALL COLOR(I,VF,HF):: NEXT I :: CALL SCREEN(BF):: RETURN
2490 !SORTIEREN
2500 CALL CLEAR
2510 VF=7 :: HF=12 :: BF=11 :: GOSUB 2480
2520 DISPLAY AT(12,4):" 1=TAGE":"    2=ARTIKELGRUPPEN"
2530 CALL KEY(0,T,S):: IF S=0 THEN 2530
2540 IF T>50 OR T<49 THEN 2530
2550 F=T-48
2560 FOR I=1 TO Z1 :: SORT=Z1-I :: FOR L=1 TO SORT
2570 IF VAL(S$(I,F))<=VAL(S$(I+L,F))THEN 2590
2580 GOSUB 2610
2590 NEXT L :: NEXT I
2600 RETURN
2610 !TAUSCH
2620 CALL CLEAR
2630 FOR K=1 TO 5 :: J$=S$(I,K):: S$(I,K)=S$(I+L,K):: S$(I+L,K)=J$ :: NEXT K
2640 RETURN
2650 END
```

*Listing »Haushaltskasse« (Schluß)*

## Haushaltskasse

Je nach Wahl wird ein Säulendiagramm auf der Basis der jeweiligen Ausgabeart erstellt. Der TI 99/4A gleicht dabei den Darstellungsmaßstab automatisch an die höchste Ausgabe des Monats an. Mit Drücken einer beliebigen Taste (außer der Leertaste) erfolgt ein Rücksprung ins Hauptmenü. Wird die Leertaste betätigt, erscheint unter dem Diagramm ein Fragezeichen. Mit den Cursortasten (ohne FCTN) kann es nach rechts oder links bewegt werden. Damit kann man es unter die Säule des Tages steuern, dessen Ausgaben besonders interessieren. Drückt man jetzt die Taste »1«, so erscheinen die Ausgaben als Liste. Mit »E« kann man unterbrechen und mit »K« die Daten korrigieren. Vor dem Rücksprung wird noch die genaue Summe am Bildschirm angezeigt.

Wählt man statt der Darstellung des gesamten Monats die einzelner Tage, kann ein bestimmter Zeitraum für eine andere Art der Säulendarstellung vorgegeben werden. Nach erfolgter Ausgabe führt die Betätigung jeder beliebigen Taste zum Hauptmenü zurück. Mit Wahl der »3« im Hauptmenü (Ausgaben) listet der Computer die Ergebnisse numerisch auf. Auch hierbei stehen die Befehle »E« für Rückkehr zum Menü und »K« für Korrekturen zur Verfügung. Die Ziffer »4« des Hauptmenüs (Sortieren) erlaubt das Sortieren der Daten. Ist der Sortiervorgang abgeschlossen, erscheint das Hauptmenü.

Sollen die Daten unter dem Programmteil 5 (Speichern) auf Kassette abgespeichert werden, muß vorher noch ein Codewort eingegeben werden. Das Programm ließe sich sicher noch um eine Reihe nützlicher Routinen erweitern. Aber dabei spielt auch die Frage nach dem vorhandenen Speicherplatz eine Rolle. Wirklich nützlich wird das Programm erst zusammen mit der Einsicht seiner Anwender, unnötige Ausgaben vermeiden zu wollen.

(H. J. Bigalke)



AndreBetz.de 95

# Solo für Sie — und Ihren TI

**Mußten Sie schon mal die Nützlichkeit Ihres TI 99/4A beweisen? Wenn Ihnen das nicht gelang, dann versuchen Sie, Ihren Kummer zu vergessen — bei einer gemütlichen Runde Solitair ganz allein zu zweit mit Ihrem Computer.**

```
5 REM PROGRAMM FUER TI99/4A
10 CALL CLEAR
20 PRINT "    S O L I T A I
 R E":::::::::
30 P$="ØØØØØØ1111ØØØØØØ"
40 CALL CHAR(121,P$)
50 CALL COLOR(12,16,16)
60 CALL HCHAR(11,1,121,32)
70 CALL HCHAR(19,1,121,32)
80 PRINT "PROGRAMMIERT VON:"
:"WOLFGANG LENK":"LANGLUETJE
NSAND 4":"2800 BREMEN 66":::
90 PRINT "BENOETIGEN SIE EIN
E         SPIELANLEITUNG?"::
"DRUECKEN SIE BITTE -J- -N-"
100 CALL KEY(Ø,K,S)
110 IF S=Ø THEN 100
120 IF K=74 THEN 140
130 IF K=78 THEN 230
140 CALL CLEAR
150 PRINT "SIE DUERFEN NUR H
ORIZONTAL  ODER VERTIKAL SPR
INGEN,UND  ZWAR MIT EINEM ST
EIN UEBER"
160 PRINT "EINEN BENACHBARTE
N STEIN AUFEIN LEERES FELD (
Z.B.IM     1.ZUG VON B4 NACH
 D4)"::
170 PRINT "DER UEBERSPRUNGEN
E STEIN    FAELLT JEDES MAL
WEG.SO IST NACH JEDEM ZUG EI
N STEIN    WENIGER IM SPIEL.
"
180 PRINT "ES WIRD ANGESTREB
T,DASS SICHDER LETZTE STEIN
NACH DEM   ZUG IM FELD D4,AL
SO IN DER"
190 PRINT "MITTE,BEFINDET.":
::"DRUECKEN SIE EINE BELIEBI
GE TASTE!"
200 CALL KEY(Ø,K,S)
210 IF S=Ø THEN 200
220 IF K 32>THEN 230
230 REM ---SPIELFELDRAND---
240 CALL CLEAR
250 CALL COLOR(5,2,16)
260 CALL COLOR(12,16,16)
270 CALL COLOR(6,2,16)
280 CALL COLOR(3,2,16)
290 CALL COLOR(4,2,16)
300 CALL COLOR(7,2,16)
310 CALL COLOR(8,2,16)
320 CALL HCHAR(1,9,120,1)
330 CALL HCHAR(17,9,120,1)
340 CALL HCHAR(1,10,65,1)
350 CALL HCHAR(17,10,65,1)
360 CALL HCHAR(1,11,120,1)
370 CALL HCHAR(17,11,120,1)
380 CALL HCHAR(1,12,66,1)
390 CALL HCHAR(17,12,66,1)
400 CALL HCHAR(1,13,120,1)
410 CALL HCHAR(17,13,120,1)
420 CALL HCHAR(1,14,67,1)
430 CALL HCHAR(17,14,67,1)
440 CALL HCHAR(1,15,120,1)
450 CALL HCHAR(17,15,120,1)
460 CALL HCHAR(1,16,68,1)
470 CALL HCHAR(17,16,68,1)
480 CALL HCHAR(1,17,120,1)
490 CALL HCHAR(17,17,120,1)
500 CALL HCHAR(1,18,69,1)
510 CALL HCHAR(17,18,69,1)
520 CALL HCHAR(1,19,120,1)
530 CALL HCHAR(17,19,120,1)
540 CALL HCHAR(1,20,70,1)
550 CALL HCHAR(17,20,70,1)
560 CALL HCHAR(1,21,120,1)
570 CALL HCHAR(17,21,120,1)
580 CALL HCHAR(1,22,71,1)
590 CALL HCHAR(17,22,71,1)
600 CALL HCHAR(1,23,120,1)
610 CALL HCHAR(17,23,120,1)
620 CALL VCHAR(1,8,120,2)
630 CALL VCHAR(1,24,120,2)
640 CALL VCHAR(3,8,49,1)
650 CALL VCHAR(3,24,49,1)
660 CALL VCHAR(4,8,120,1)
670 CALL VCHAR(4,24,120,1)
680 CALL VCHAR(5,8,50,1)
690 CALL VCHAR(5,24,50,1)
700 CALL VCHAR(6,8,120,1)
710 CALL VCHAR(6,24,120,1)
720 CALL VCHAR(7,8,51,1)
730 CALL VCHAR(7,24,51,1)
740 CALL VCHAR(8,8,120,1)
750 CALL VCHAR(8,24,120,1)
760 CALL VCHAR(9,8,52,1)
770 CALL VCHAR(9,24,52,1)
780 CALL VCHAR(10,8,120,1)
790 CALL VCHAR(10,24,120,1)
800 CALL VCHAR(11,8,53,1)
810 CALL VCHAR(11,24,53,1)
820 CALL VCHAR(12,8,120,1)
830 CALL VCHAR(12,24,120,1)
840 CALL VCHAR(13,8,54,1)
850 CALL VCHAR(13,24,54,1)
860 CALL VCHAR(14,8,120,1)
870 CALL VCHAR(14,24,120,1)
880 CALL VCHAR(15,8,55,1)
890 CALL VCHAR(15,24,55,1)
900 CALL VCHAR(16,8,120,1)
910 CALL VCHAR(16,24,120,1)
920 CALL VCHAR(17,8,120,1)
930 CALL VCHAR(17,24,120,1)
940 REM ---SPIELFELD---
950 CALL CHAR(96,"3C7EFFFFFF
FF7E3C")
960 CALL COLOR(9,2,1)
970 CALL HCHAR(22,1,32,32)
980 FOR Z=3 TO 15 STEP 2
990 FOR X=14 TO 18 STEP 2
1000 CALL HCHAR(Z,X,96,1)
1010 NEXT X
1020 NEXT Z
1030 FOR Z=7 TO 11 STEP 2
1040 FOR S=10 TO 12 STEP 2
```

*Listing »Solitair«*



AndreBetz.de 50

```
1050 CALL VCHAR(Z,S,96,1)
1060 NEXT S
1070 NEXT Z
1080 FOR Z=7 TO 11 STEP 2
1090 FOR S=20 TO 22 STEP 2
1100 CALL VCHAR(Z,S,96,1)
1110 NEXT S
1120 NEXT Z
1130 CALL COLOR(2,2,1)
1140 CALL HCHAR(9,16,43,1)
1150 EIN$="VON    NACH    "
1160 FOR K=1 TO 15
1170 CALL HCHAR(21,K+7,ASC(S
EG$(EIN$,K,1)))
1180 NEXT K
1190 P=32
1200 REM ---KOORDINATENEINGA
BEN---
1210 CALL HCHAR(21,12,32,2)
1220 CALL HCHAR(21,20,32,2)
1230 CALL KEY(Ø,KEY,STATUS)
1240 IF STATUS=Ø THEN 1230
1250 IF KEY<65 THEN 1230
1260 IF KEY>71 THEN 1230
1270 V$=CHR$(KEY)
1280 CALL HCHAR(21,12,KEY,1)
1290 CALL KEY(Ø,KEY,STATUS)
1300 IF STATUS=Ø THEN 1290
1310 IF KEY<49 THEN 1290
1320 IF KEY>55 THEN 1290
1330 V1$=CHR$(KEY)
1340 CALL HCHAR(21,13,KEY,1)
1350 CALL KEY(Ø,KEY,STATUS)
1360 IF STATUS=Ø THEN 135C
1370 IF KEY<65 THEN 1350
1380 IF KEY>71 THEN 1350
1390 N$=CHR$(KEY)
1400 CALL HCHAR(21,20,KEY,1)
1410 CALL KEY(Ø,KEY,STATUS)
1420 IF STATUS=Ø THEN 1410
1430 IF KEY<49 THEN 1410
1440 IF KEY>55 THEN 1410
1450 N1$=CHR$(KEY)
1460 CALL HCHAR(21,21,KEY,1)
1470 REM ---KONTROLLE AUF KO
RREKTEN ZUG---
1480 A=ASC(V$)
1490 B=ASC(V1$)
1500 C=ASC(N$)
1510 D=ASC(N1$)
1520 IF A=65 THEN 1560
1530 IF A=66 THEN 1560
1540 IF A=70 THEN 1560
1550 IF A=71 THEN 1560 ELSE
1600
1560 IF B=49 THEN 1210
1570 IF B=50 THEN 1210
1580 IF B=54 THEN 1210
1590 IF B=55 THEN 1210 ELSE
1600
1600 IF C=65 THEN 1640
1610 IF C=66 THEN 1640
1620 IF C=70 THEN 1640
1630 IF C=71 THEN 1640 ELSE
1680
1640 IF D=49 THEN 1210
1650 IF D=50 THEN 1210
1660 IF D=54 THEN 1210
1670 IF D=55 THEN 1210 ELSE
1680
1680 IF D=B-2 THEN 1690 ELSE
1700
1690 IF A=C THEN 1770 ELSE 1
210
1700 IF D=B+2 THEN 1710 ELSE
 1720
1710 IF A=C THEN 1770 ELSE 1
210
1720 IF C=A-2 THEN 1730 ELSE
 1740
1730 IF B=D THEN 1770 ELSE 1
210
1740 IF C=A+2 THEN 1750 ELSE
 1210
1750 IF B=D THEN 1770 ELSE 1
210
1760 REM ---1.FELD FREI MACH
EN---
1770 S=8
1780 FOR I=65 TO 71
1790 S=S+2
1800 IF A=I THEN 1820
1810 NEXT I
1820 Z=1
1830 FOR I=49 TO 55
1840 Z=Z+2
1850 IF B=I THEN 1870
1860 NEXT I
1870 CALL HCHAR(Z,S,43,1)
1880 CALL SOUND(100,294,Ø)
1890 REM ---2.FELD FREI MACH
EN---
1900 IF A>C THEN 1910 ELSE 1
930
1910 CALL HCHAR(Z,S-2,43,1)
1920 CALL SOUND(100,294,Ø)
1930 IF A<C THEN 1940 ELSE 1
960
1940 CALL HCHAR(Z,S+2,43,1)
1950 CALL SOUND(100,294,Ø)
1960 IF B>D THEN 1970 ELSE 1
990
1970 CALL HCHAR(Z-2,S,43,1)
1980 CALL SOUND(100,294,Ø)
1990 IF B<D THEN 2000 ELSE 2
020
2000 CALL HCHAR(Z+2,S,43,1)
2010 CALL SOUND(100,294,Ø)
2020 REM ---FELD BELEGEN---
2030 S=8
2040 FOR I=65 TO 71
2050 S=S+2
2060 IF C=I THEN 2080
2070 NEXT I
2080 Z=1
2090 FOR I=49 TO 55
2100 Z=Z+2
2110 IF D=I THEN 2130
2120 NEXT I
2130 CALL HCHAR(Z,S,96,1)
2140 CALL SOUND(100,220,Ø)
2150 P=P-1
2160 IF P=1 THEN 2170 ELSE 1
210
2170 IF C=68 THEN 2180
2180 IF D=52 THEN 2190 ELSE
2400
2190 CALL HCHAR(20,1,32,32)
2200 FOR A=1 TO 10
2210 CALL SOUND(-100,660,Ø)
2220 NEXT A
2230 FOR A=1 TO 10
2240 CALL SOUND(-100,660,Ø,5
50,Ø)
2250 NEXT A
2260 FOR A=1 TO 10
2270 CALL SOUND(-100,660,Ø,5
50,Ø,440,Ø)
2280 NEXT A
2290 FOR A=Ø TO 30 STEP 2
2300 CALL SOUND(-100,660,A,5
50,A,440,A)
2310 NEXT A
2320 HERZ$="HERZLICHEN GLUEC
KWUNSCH"
2330 CALL HCHAR(21,1,32,32)
2340 FOR K=1 TO 23
2350 CALL HCHAR(21,K+3,ASC(S
EG$(HERZ$,K,1)))
2360 NEXT K
2370 CALL HCHAR(21,14,120,1)
2380 CALL HCHAR(21,27,120,4)
```

*Listing »Solitair« (Fortsetzung)*



AndreBetz.de 51

```
2390 GOSUB 2510
2400 CALL HCHAR(21,1,32,32)
2410 CALL HCHAR(21,4,121,27)
2420 SIE$="ES GEHT NICHT MEH
R WEITER"
2430 FOR K=1 TO 25
2440 CALL HCHAR(21,K+3,ASC(S
EG$(SIE$,K,1)))
2450 NEXT K
2460 CALL HCHAR(21,6,120,1)
2470 CALL HCHAR(21,11,120,1)
2480 CALL HCHAR(21,17,120,1)
2490 CALL HCHAR(21,22,120,1)
2500 GOSUB 2510
2510 REM ---SUBROUTINE---
2520 CALL HCHAR(22,4,120,27)
2530 NEU$="WOLLEN SIE EIN NE
UES SPIEL?"
2540 DRU$="DRUECKEN SIE -J-
-N-"
2550 FOR L=1 TO 27
2560 CALL HCHAR(23,L+3,ASC(S
EG$(NEU$,L,1)))
2570 NEXT L
2580 CALL HCHAR(23,10,120,1)
2590 CALL HCHAR(23,14,120,1)
2600 CALL HCHAR(23,18,120,1)
2610 CALL HCHAR(23,24,120,1)
2620 CALL HCHAR(24,4,121,27)
2630 FOR M=1 TO 20
2640 CALL HCHAR(24,M+3,ASC(S
EG$(DRU$,M,1)))
2650 NEXT M
2660 CALL HCHAR(24,12,120,1)
2670 CALL HCHAR(24,16,120,1)
2680 CALL HCHAR(24,20,120,1)
2690 CALL KEY(0,K,S)
2700 IF S=0 THEN 2690
2710 CALL HCHAR(21,1,32,32)
2720 CALL HCHAR(23,1,32,32)
2730 CALL HCHAR(24,1,32,32)
2740 IF K=74 THEN 950
2750 IF K=78 THEN 2760
2760 CALL CLEAR
2770 STOP
```

*Listing »Solitair« (Schluß)*

***Für dieses Spiel wird nur das Grundgerät benötigt***

Das Programm wird mit "RUN" und "ENTER" gestartet. Es besteht am Anfang aus 32 Steinen. Es darf nur horizontal oder vertikal gesprungen werden, und zwar mit einem Stein über einen benachbarten Stein auf ein leeres Feld (zum Beispiel im 1. Zug von B4 nach D4). Jeder übersprungene Stein verschwindet vom Spielfeld. So ist nach jedem Zug ein Stein weniger im Spiel. Unser Ziel ist, daß sich der letzte Stein nach dem letzten Zug im Feld D4, also in der Mitte, befindet.

Das Spiel ist nicht einfach. Sie werden viel Geduld brauchen. Es ist damit genau das richtige Spiel für lange Herbst- oder Winterabende.

(Wolfgang Lenk)

# Sternjäger: Düst im Sauseschritt

**Mit seinem Sternjäger schwirrt man durch die Unendlichkeit des Alls und besucht fremde Welten. In acht nehmen sollte man sich vor den Neutronensternen und der eigenen Reiseroute.**

Commodore 64

Das Spiel »Sternjäger« für den Commodore 64 ist komplett in Basic geschrieben. Obwohl Basic eine langsame Sprache ist, konnte das Programm durch Optimierung der zeitkritischen Programmteile doch relativ schnell gehalten werden.

Zunächst einmal zum Spielablauf selbst. Die Aufgabe des Spielers besteht darin, über den Bildschirm zu düsen und in der Zeit von 90 Sekunden möglichst viele Sterne einzusammeln. Die eingesammelten Sterne folgen dabei dem Sternenjäger als Schlange, wobei jeder neu hinzukommende Stern die Schlange um ein

```
BA              : Basisadresse des Bildschirm-RAM
BO              : Bonuswert
BZ              : Zwischenwert fuer Cursorsteuerung
BO,B1,B2,B4,B8  : Variable fuer Bitoperationen bei der Joystickabfrage
C1, C2, C3      : ASCII-Code fuer gefangenen Stern, Sternjaeger, Stern
CP              : Adresse von Controlport 1
E, L            : Pointer auf erstes und letztes Element der Sternschlange
FR              : Zustand der Feuertaste (1=gedrueckt, 0=nicht gedrueckt)
G (*)           : Array fuer Positionen der Sternschlangenelemente
GS              : Geschwindigkeit des Sternjaegers
HI              : Highscore
I, J            : Schleifenzaehler
IX              : Indikator fuer Spielzustand
IS, IZ          : Spalte und Zeile fuer Cursor-Positionierung
LE              : ASCII-Code fuer Leerzeichen
MX              : Groesse des Arrays G (*)
PU              : Punktestand des momentanen Spieles
```

*Variablendefinition*


AndreBetz.de 52

# TI 99/4A durchsichtig gemacht

**Besitzer des TI 99/4A sind gegenüber den Besitzern anderer populärer Heimcomputer in einem Punkt etwas im Nachteil: Da die interne Rechnerarchitektur der CPU TMS9900 sehr stark von derjenigen gängiger CPUs abweicht, stellt allgemeinere Literatur zur Maschinencode-Programmierung für TI-Fans keine Hilfe dar. Texas Instruments publiziert allerdings eine Reihe Bücher zur Programmierung des TMS 9900, die im Buchhandel (meist erst auf Bestellung) zu erhalten sind.**

Eine relativ kurze Einführung bietet zum Beispiel der Band »TM 990/100MA, Mikrocomputer Anwenderhandbuch«. Zwar bezieht sich dieses Lehrbuch unmittelbar auf den Prozessor TM990, und das damit aufgebaute Einplatinenlehrsystem TM990/100MA. Dieser Prozessor ist aber kompatibel zum TMS 9900. Der Nachteil des Buches für den TI 99/4A-Anwender ist, daß eine Kenntnis des Befehlssatzes des Prozessors alleine beim TI 99/4A wegen seiner komplizierten RAM-Organisation für die Praxis nichts hilft. Wer sich aber das Lehrsystem zusätzlich zulegt, findet einen Einstieg, um den ihn jeder Nur-Heimcomputer-Besitzer beneiden wird. (200 Seiten; 17,50 Mark)

Ausführlicher und auch stärker auf die Bedürfnisse des TI 99/4A-Anwenders abgestimmt, ist das Buch »16-Bit-Mikroprozessor Kursbuch«, das jeden einzelnen Befehl des Maschinencodes sehr übersichtlich und verständlich darstellt. Auch dieses Buch liegt den Betrachtungen das oben erwähnte Lehrsystem zugrunde, widmet aber auch einige Kapitel allgemeinen Themen, wie zum Beispiel Softwareentwicklung und Produktentwicklung. An diesen Themen kann man auch ablesen, daß das Buch nicht für den Hobbybereich geschrieben wurde. Dennoch muß man zum Lesen kein Profi sein. (460 Seiten; 32,50 Mark).

## Manches gibt's leider nur in Englisch

Beide Bücher können aber nur theoretische Kenntnisse vermitteln oder vorhandenes ergänzen. Die Grundlage zur Maschinencode-Programmierung des TI 99/4A vermittelt das Handbuch zum Editor/Assembler-Paket. Dieses umfangreiche Lehrmaterial ist gegenwärtig allerdings nur in Englisch erhältlich und leider nicht gerade leicht zu lesen (auch bei guten Sprachkenntnissen), da durch die besondere Prozessor-Architektur eine Reihe von Fachausdrücken recht eigenwillig verwendet werden. Wichtig ist für den Anwender, daß dieses Handbuch von Texas Instruments auch ohne den Editor/Assembler bezogen werden kann. Es kostet dann zirka 40 Mark im Buchhandel.

Zwei von Texas Instruments angekündigte Neuerscheinungen sind in dieser Hinsicht besonders interessant. Anfang November soll ein Band mit dem Titel »99er Special I« ausgeliefert werden, der zum Preis von 49,50 Mark eine Einführung in die Programmiersprachen Basic, Extended Basic, Assembler und UCSD Pascal enthält. Allerdings ist dieses Buch trotz vieler Programmbeispiele nicht für den reinen Anfänger gedacht, da eine Reihe von Vorkenntnissen in Basic vorausgestzt wird.

Die zweite Neuerscheinung nennt sich »Mini Memory/Editor Assembler«, wird zirka 50 Mark kosten und soll gegen Ende des Jahres im Buchhandel erhältlich sein. Inhalt ist eine lehrhafte Einführung in den Gebrauch dieser beiden Assemblerwerkzeug-Pakete.

Wer sich als Anwender und Fan des TI 99/4A gerne laufend über Neuheiten informieren lassen möchte, ist gut beraten, das »99er Home Computer Magazine« von Texas Instruments USA zu abonnieren. Bisher ausschließlich in Englisch erhältlich, soll es in Zukunft auch eine (originale) deutsche Ausgabe geben. Der Preis ist 43 US $ pro Jahr bei Versendung mit dem Schiff. Der Inhalt ist seinen Preis tatsächlich wert. Neben vielen Informationen zum Programmieren enthält jedes Heft mehrere Listings von nützlichen und Spielprogrammen. Manches daraus abgetippte Programm ist besser als kommerziell vertriebene Software. Für das Abonnement muß man sich an 99er Home Computer Magazine, P.O. Box 5537, Eugene, OR 97405 wenden. Die Bezahlung sollte am besten mit einem Bankscheck auf US $ lautend erfolgen, den man selber an die angegebene Adresse unter Angabe des Absenders und Verwendungszweck schickt. Wer eine Kreditkarte hat, kann bei der Bestellung noch angeben, man solle sein Konto Nummer.... belasten.

Sämtliche Anweisungen, Unterprogramme und Funktionen des TI Basic und TI Extended Basic erklärt das Buch »TI Basic/Extended Basic für Anfänger und Fortgeschrittene« von Texas Instruments anschaulich und ausführlich.

Das Buch ist in zwei Teile gegliedert. Der erste führt den Leser schrittweise an das Programmieren in Basic heran. Es wird gezeigt, wie man den Computer benutzt oder einfache Programme mit dem Editor schreibt. Der Leser wird angeleitet grafische Muster in 16 Farben auf dem Bildschirm zu entwerfen sowie Töne, Melodien und Akkorde auszugeben. Im zweiten Teil werden sämtliche Befehle alphabetisch geordnet dargestellt. Damit ermöglicht dieses Buch nicht nur einen ersten Zugang zum erweiterten Basic, sondern wird darüber hinaus dem bereits geübten Benutzer zu einem übersichtlichen Nachschlagewerk.

Für jeden Befehl wird zuerst das Format angegeben. Dann wird er ausführlich beschrieben und an einem Beispiel erläutert. In den meisten Fällen handelt es sich bei dem Beispiel um ein vollständiges Programm.

Das Buch ist jedem zu empfehlen, der in TI Extended Basic programmiert und die Fähigkeiten dieser Sprache voll ausschöpfen möchte.


AndreBetz.de 110

# Bildschirmroutinen für die totale Kontrolle

**Diese Poke-Liste für den TI 99/4A mit dem Minimem-Modul stattet das TI-Basic mit einer Reihe neuer Befehle aus, die recht ungewöhnliche Effekte am Bildschirm erlauben und weit über die bisherigen Möglichkeiten hinausgehen.**

Der vom Programm belegte Speicherraum liegt unterhalb der Adresse >7D00, ab der im Minimem üblicherweise Maschinenprogramme zu schreiben sind. Das heißt: »Bildschirm total« kann mit eigenen, zusätzlichen Routinen kombiniert werden, sofern darauf bei der Erstellung der Namensliste bis zur Adresse >7FC0 Rücksicht genommen wird und die neuen Adressen von FFAM (Systemvariable) und dem Beginn der Namenstabelle in Adresse >701C eventuell korrigiert werden (siehe Listing, Programmzeile 880).

Eine Kurzbeschreibung der acht neuen Befehle wird in den REM-Zeilen des Programms gegeben, die Scroll-Befehle für das Verschieben des Bildschirms nach links, rechts und unten bewirken pro Aufruf jeweils eine Verschiebung um eine Spalte beziehungsweise Zeile bei gleichzeitigem Auffüllen der neugewonnenen Spalte oder Zeile mit Leerzeichen.

Der Befehl »CALL LINK ("MASKE",Z1,Z2,S1,S2,DIR)« erlaubt ein Scrollen des Bildschirms in eine vom Wert der Variablen DIR abhängige Richtung, wobei ein Bildschirmfenster zwischen den Zeilen Z1,Z2 und den Spalten S1,S2 in Ruhe bleibt. Diese Routine kann zum Beispiel eingesetzt werden, um, unabhängig von irgendwelcher Bildschirmgrafik, Erläuterungen zur Eingabe am Schirm auszugeben.

## *Ruhende Felder — bewegte Felder: MASKE und WINDOW*

Die Umkehrung dieses Befehls ist in »CALL LINK ("WINDOW",Z1,Z2,S1,S2,DIR)« zu sehen, wobei bei unverändertem Bildschirm ein durch die Zeilen- und Spaltenangabe definierter Bildschirmabschnitt aufwärts oder abwärts gescrollt werden kann. Auch diese Routine kann zum Beispiel dazu benutzt werden, eine Bildschirmgrafik zu schonen und dennoch variable Texte einzuspielen.

»CALL LINK("HIDE")« und »CALL LINK("SEEK")« dienen zur Abspeicherung eines kompletten Bildschirms (da auch die Farb- und Patterntabelle mitgespeichert wird, darf auch Grafik am Bildschirm sein). Diese Routine ist dann nützlich, wenn rasch große Veränderungen am Bildschirm stattfinden sollen und bestimmte Bildschirmmuster häufiger im Programm benötigt werden. Durch das Mitspeichern der Farb- und Patterntabelle ist es auf diesem Weg möglich, zur gleichen Zeit einen Charakter sowohl mit anderem Grafikmuster als auch anderer Farbgebung als der gerade definierten im Gerät präsent zu haben.

»CALL LINK("INVERS",ASCII1,ASCII2)« letztlich ist bei viel Text am Bildschirm nützlich, weil einzelne Zeichen invers dargestellt werden können und damit eine optische Hervorhebung von Abschnitten leichter möglich ist; eine Grafikmöglichkeit, die am TI sonst durch die notwendige Umdefinition von 8er-Blöcken sehr umständlich zu realisieren und langsam ist.

Es wäre auch möglich, zumindest vor dem ersten Programmaufruf von »HIDE« den Speicherraum >7118 bis >7418 noch mit dem bereits in der Ausgabe 11/83 publizierten Programm »SMALL« aufzufüllen und dieses vor dem ersten Aufruf von »HIDE« anzuwählen, damit noch die Kleinbuchstaben hinzukommen. Allerdings: Im Falle eines Programmabbruchs können dann die Kleinbuchstaben nicht mehr mit »CALL LINK("SMALL")« aufgerufen werden, wenn inzwischen »HIDE« verwendet wurde.

(Listing siehe Seite 46)

(Karl Hagenbuchner)

## Idee sucht Ausführung

**Manche gute Programm-Idee wartet ein Leben lang darauf, ausgeführt zu werden...**
**...und mancher Programmierer wartet ein Leben lang auf eine gute Idee. Wir wollen beide zusammenbringen.**

Vielleicht haben gerade Sie die Programmidee, nach der alle Welt sucht, aber keine Zeit für die Ausarbeitung. Diese Idee wollen wir unkommentiert in einer Ideenliste mit Namen und Adresse veröffentlichen.

Vielleicht haben Sie gerade keine Idee, was Sie programmieren könnten, aber es juckt Ihnen in den Fingern und Zeit wäre auch gerade. Wenn Sie sich für eine dieser Ideen interessieren, schauen Sie in Zukunft in unsere Ideenecke. Sollten Sie dann noch weitere Informationen für die Umsetzung in ein Programm brauchen, wenden Sie sich bitte an den jeweiligen Ideenlieferanten.

Außerdem gibt es ein paar **Spielregeln**, die jeder, der an dieser Aktion teilnimmt, aus Gründen der Fairniß einhalten sollte (wir sind übrigens der Meinung, daß alle unsere Leser fair sind und überwachen die Einhaltung nicht!).

### Wer eine Idee hat

- ★ schickt uns diese Idee, mit oder ohne nähere Erläuterung, auf einer Karte oder in einem Brief, mit Namen und Adresse versehen
- ★ erhält im Falle einer erfolgreichen Ausarbeitung durch einen anderen Leser, wenn dessen Programm in Happy-Computer veröffentlicht wird, für die Idee ein Honorar über 50 Mark

### Wer eine Idee aufgreift

- ★ und in ein Programm einbaut, erwähnt neben dem eigenen Namen und der eigenen Adresse den Namen und die Adresse des Ideenanbieters in der Kopfzeile des Programms oder der entsprechenden Subroutine
- ★ schickt eine Kopie des Programms an den Ideenanbieter (möglichst auf Datenträger) zur freien Verwendung
- ★ schickt an die Redaktion entweder ein gut lesbares Listing mit Programmbeschreibung (2zeilig, mit 50 Zeichen pro Zeile) zur Veröffentlichung (Honorar mindestens 100 Mark)
- ★ oder wenigstens eine Erfolgsnachricht (in diesem Fall zahlt die Redaktion dem Ideenanbieter kein Honorar und es ist Ehrensache, daß derjenige, der so ein Programm kommerziell verwertet, den Urheber der Idee am Gewinn beteiligt)

**Einsendungen an**
**Happy-Computer,**
**Aktion Ideenecke,**
**Hans-Pinsel-Str. 2,**
**8013 Haar bei München**

Die Redaktion übernimmt für etwaige Folgen einer Ideenverwertung keine Haftung

```
100 REM BILDSCHIRM-TOTAL: Maschinensprachenprogramme zur
110 REM                   Bildschirmgestaltung
120 REM
130 REM (c) Mag.Karl Hagenbuchner,Haidfeldstr. 2, A-4850 Traun
140 REM Diese Pokeliste erzeugt ein Maschinenprogramm, mit dessen Hilfe
150 REM folgende Routinen ermöglicht werden:
160 REM  CALL LINK("LINKS")     scrollt um eine Spalte nach links
170 REM  CALL LINK("RECHTS")    scrolit um eine reihe nach rechts
180 REM  CALL LINK("DOWN")      scrollt um eine Zeile herunter
190 REM  CALL LINK("HIDE")      speichert einen Bildschirm
200 REM  CALL LINK("SEEK")      bringt gespeicherten Bildschirm
210 REM  CALL LINK("WINDOW",Z1,Z2,S1,S2,DIR)
220 REM                         scrolit einen Bildschirmabschnitt zw. Zeile Z1
230 REM                         und Zeile Z2, Spalte1 und Spalte2 hinauf (DIR=0)
240 REM                         oder hinunter (DIR=1)
250 REM  CALL LINK("MASKE",Z1,Z2,S1,S2,DIR)
260 REM                         scrollt den Schirm bei ruhender Maske, begrenzt
270 REM                         durch Zeile1, Zeile 2, Spalte1, Spalte2
280 REM                         nach links (DIR=0), rechts (DIR=1),
290 REM                         hinunter (DIR=2) und hinauf (DIR=3)
300 REM  CALL LINK("INVERS",ASCII1,ASCII2)
310 REM                         setzt einzelne Charakters (ASCII 1) invers und
320 REM                         definiert damit einen zweiten Charakters
330 REM                         Rückbezug ist möglich: ASCII 1 = ASCII 2
340 REM ************************************************************************
350 REM Dieses Programm läuft nur mit MINIMEM !
360 REM ************************************************************************
370 CALL INIT
380 DATA 194,139,6,160,121,162,5,129,5,133,4,32,96,40,2,32,0,31,192
390 DATA 68,4,32,96,36,5,128,2,37,0,32,192,69,6,3,22,243,4,90,194
400 DATA 139,6,160,121,162,5,128,4,32,96,40,6,0,192,68,4,32,96,36,2
410 DATA 32,0,32,2,37,0,32,192,69,6,3,22,242,4,90,4,192,2,1,118,56
420 DATA 2,2,3,0,4,32,96,48,193,65,2,4,128,0,2,3,0,24,2,2,0,31,4
430 DATA 91,4,192,2,1,118,56,2,2,2,224,4,32,96,48,2,32,0,32,4,32,96
440 DATA 40,2,1,128,0,2,2,0,32,4,192,4,32,96,36,5,128,6,2,22,251,4
450 DATA 91,194,139,6,160,122,36,4,32,96,48,6,160,122,48,4,32,96,48,6
460 DATA 160,122,62,4,32,96,48,4,90,194,139,6,160,122,36,4,32,96,40,6
470 DATA 160,122,48,4,32,96,40,6,160,122,62,4,32,96,40,4,90,4,192,2
480 DATA 1,113,24,2,2,3,0,4,91,2,0,3,0,2,1,116,24,2,2,0,32,4,91,2
490 DATA 0,4,0,2,1,116,56,2,2,2,0,4,91
500 DATA 4,32,96,68,4,32,96,28,18,0,142,224,131,74,17,4,142,224,131
510 DATA 74,21,1,4,91,2,0,19,0,4,32,96,80,194,139,4,192,2,1,0,1,6
520 DATA 160,122,76,0,1,0,23,193,160,131,74,5,129,6,160,122,76,0,2,0
530 DATA 24,129,160,131,74,17,234,193,224,131,74,5,129,6,160,122,76,0
540 DATA 1,0,31,194,32,131,74,5,129,6,160,122,76,0,2,0,32,130,32,131
550 DATA 74,17,217,194,96,131,74,5,129,6,160,122,76,0,0,0,3,195,96
560 DATA 131,74,4,90,195,11,6,160,122,108,6,6,10,86,161,136,5,134,6,7
570 DATA 10,87,161,200,5,135,5,137,98,72,2,141,0,0,19,37,2,141,0,1
580 DATA 22,188,192,7,2,32,0,32,2,1,121,56,192,137,2,32,255,192,4,32
590 DATA 96,48,129,128,19,5,2,32,0,32,4,32,96,40,16,245,2,32,0,32,4
600 DATA 32,96,40,2,32,255,224,2,15,128,0,220,79,6,2,22,253,2,1,121
610 DATA 56,192,137,4,32,96,40,4,92,192,6,2,32,255,224,2,1,121,56,192
620 DATA 137,2,32,0,64,4,32,96,48,2,32,255,224,129,192,17,9,19,1,4
630 DATA 92,195,65,195,130,2,15,128,0,223,79,6,14,22,253,4,32,96,40,16,236
640 DATA 194,139,4,192,2,1,0,1,6,160,123,192,192,224,131,74,5,129,6
650 DATA 160,123,192,193,32,131,74,2,0,3,248,2,5,0,31,2,32,0,8,6,3
660 DATA 129,67,22,251,2,1,121,56,2,2,0,8,4,32,96,48,2,6,121,56,2,3
670 DATA 0,4,5,22,6,54,6,3,22,252,2,0,3,248,2,32,0,8,6,4,129,68,22
680 DATA 251,4,32,96,40,4,90,4,32,96,68,4,32,96,28,18,0,4,91
690 DATA 195,11,6,160,122,108,195,77,19,7,2,141,0,1,19,7,2,141,0,2
700 DATA 19,7,16,11,6,160,121,88,16,34,6,160,121,126,16,31,6,160,121
710 DATA 162,6,160,121,192,16,26,6,160,121,162,2,0,0,32,2,2,0,32,4
720 DATA 32,96,48,2,32,255,224,4,32,96,40,2,32,0,64,2,128,3,0,17,245
730 DATA 2,0,2,224,2,1,128,0,4,32,96,36,5,128,6,2,22,251,6,160,124
```

*Listing für die totale Bildschirmkontrolle beim Scrollen mit dem TI 99/4A*

# TI 99/4A — uhriger Computer

**Mit diesem Programm wird Ihr TI 99/4A zu einer universellen und sehr genau gehenden Uhr, Stoppuhr, oder einem Wecker**

```
10 WM=99
20 CALL SCREEN(2)
30 CALL CLEAR
40 PRINT "DER TI-99/4 ALS UNIVER
SALUHR":::"WELCHEN PROGRAMMTEIL?
???"::
50 PRINT "-STOPPUHR   (TASTE 1)"
:"-NORMALE UHR(TASTE 2)":"-WECKE
R     (TASTE 3)":::
60 GOSUB 850
70 CALL KEY(0,KEY,S1)
80 IF S1=0 THEN 70
90 IF KEY=49 THEN 290
100 IF KEY=50 THEN 220
110 IF KEY<>51 THEN 70
120 PRINT "WECKEN MIT SPRACHGENE
RATOR??DRUECKEN SIE 'J' ODER 'N'
 !!":::
130 CALL SOUND(150,1400,5)
140 CALL KEY(0,K,S1)
150 IF S1=0 THEN 140
160 IF K=78 THEN 200
170 IF K<>74 THEN 130
180 SPRACHE=1
190 CALL SAY("ALL+RIGHT...ENTER
TIME")
200 INPUT "GEBEN SIE DIE WECKZEI
T EIN  (STUNDEN;MINUTEN). ":WS,W
M
210 PRINT
220 PRINT "GEBEN SIE NUN BITTE":
"DIE AKTUELLE ZEIT EIN.":::
230 INPUT "STUNDEN=? ":ST
240 INPUT "MINUTEN=? ":M
250 A=INT(ST/10)
260 B=ST-A*10
270 C=INT(M/10)
280 D=M-C*10
290 CALL CLEAR
300 IF KEY=51 THEN 900
310 CALL COLOR(9,2,2)
320 PRINT "DER TI-99/4 ALS UNIVE
RSALUHR":::"VON":" TORSTEN NIEMIE
TZ":" MARBACHER WEG 39":" D-2800
 BREMEN 1":::::
330 PRINT ::::::::::::
340 CALL HCHAR(12,12,96,10)
350 CALL HCHAR(14,12,96,10)
360 CALL HCHAR(13,12,96)
370 CALL HCHAR(13,21,96)
380 CALL HCHAR(13,15,58,4)
390 CALL HCHAR(13,13,A+48)
400 CALL HCHAR(13,14,B+48)
410 CALL HCHAR(13,16,C+48)
420 CALL HCHAR(13,17,D+48)
430 CALL HCHAR(13,19,48,2)
440 CALL KEY(0,K,S1)
450 IF S1=0 THEN 440
460 IF M<>WM THEN 530
470 IF ST<>WS THEN 530
480 GOSUB 850
490 IF SPRACHE<>1 THEN 510
500 CALL SAY("HELLO,IT+IS TIME T
O GET UP")
510 GOSUB 940
520 GOTO 480
530 CALL SOUND(820,44733,30)
540 CALL KEY(0,K,S1)
550 IF S1=0 THEN 600
560 IF K<>83 THEN 600
570 GOSUB 940
580 CALL SOUND(800,262,0)
590 GOTO 590
600 IF SE=59 THEN 670
610 SE=SE+1
620 E=INT(SE/10)
630 F=SE-E*10
640 CALL HCHAR(13,19,E+48)
650 CALL HCHAR(13,20,F+48)
660 GOTO 460
670 IF M=59 THEN 750
680 M=M+1
690 C=INT(M/10)
700 D=M-C*10
710 CALL HCHAR(13,16,C+48)
720 CALL HCHAR(13,17,D+48)
730 SE=0
740 GOTO 620
750 IF ST=23 THEN 830
760 ST=ST+1
770 A=INT(ST/10)
780 B=ST-A*10
790 CALL HCHAR(13,13,A+48)
800 CALL HCHAR(13,14,B+48)
810 M=0
820 GOTO 690
830 ST=0
840 GOTO 770
850 FOR SOUND=1 TO 16
860 CALL SCREEN(SOUND)
870 CALL SOUND(100,SOUND*131,0)
880 NEXT SOUND
890 RETURN
900 CALL SCREEN(2)
910 CALL COLOR(3,14,1)
920 CALL COLOR(4,14,1)
930 GOTO 380
940 FOR TERZ=1 TO 20
950 CALL SOUND(100,523,0)
960 CALL SOUND(100,659,0)
970 NEXT TERZ
980 RETURN
```

*Listing »Universaluhr«*

Diese Uhr geht normalerweise auch nach mehrstündigem Einsatz keine Sekunde falsch. Sollte es aber dennoch einmal bei einem Benutzer dieses Programms zu Zeitdifferenzen kommen, so muß er die erste Zahl in Programmzeile 530 abändern.

### *Wecken mit Fanfare und Sprachausgabe*

Und so muß das Programm bedient werden:
☐ Programm einlesen oder eintippen,
☐ RUN und ENTER drücken. Das Menü erscheint.
☐ Drücken Sie nun die 1 für Stoppuhr. Sie können die Uhr mit einem Druck auf eine beliebige Taste starten und mit einem weiteren Druck auch wieder stoppen.
Oder
☐ Sie drücken 2 für »normale Uhr«. Befolgen Sie bitte die Bildschirminstruktionen.
Oder
☐ Sie drücken 3 für Wecker (wahlweise mit oder ohne Sprachgenerator). Da Ihr TI 99/4A nun als Wecker dienen soll, erfolgt eine Dunkelschaltung. Geweckt werden Sie mit einer Reihe von Tönen, gefolgt von einer Fanfare sowie mit einem Satz, falls Sie den Sprachgenerator besitzen.
☐ Beendet wird das Programm mit »SHIFT C/SHIFT Q« oder für die TI 99/4A-Besitzer mit FCTN 4 oder FCTN 9.

Dieses Programm simuliert übrigens eine 24-Stunden-Uhr, das heißt, daß bei Erreichen der Uhrzeit 23.59.59 automatisch auf 0.00.00 Uhr umgesprungen wird (und zwar ohne Zeitverlust).

(Torsten Niemietz)

# Der TI 99/4A als Musikus

**Mit dem Programm »Musik-Lehrer« in TI-Basic können Sie Ihr Gehör schulen und testen. Damit werden Sie zwar noch kein Beethoven aber taub.**

Das Programm erzeugt Töne der C-Dur-Tonleiter, die man erkennen muß. Man schreibt sich ein Notendiktat oder läßt sich Tonintervalle vorspielen. Einfach? Ist Ihr Trommelfell trainiert? Ein Tip: Nur die Töne c,d,e, f,g,a,h werden verwendet.
☐ Bedienung:
1. Tippen oder lesen Sie das Programm von Kassette ein.
2. Während des Vorspanns haben Sie nun die Möglichkeit, für die Menüwahl die Tasten mit den Ziffern 1 bis 3 zu benutzen.
3. Haben Sie auf die 1 gedrückt, so haben Sie sich für das Notendiktat entschieden. Als erstes hören Sie den Ton C. Geben Sie »C« ein. Welchen Ton hören Sie jetzt? Geben Sie den Notennamen ein. Es empfiehlt sich, mit der Eingabe nicht zu lange zu zögern, damit einem der Ton noch im Ohr liegt. Der Computer springt nie mehr als 2 Töne. Dies können Sie aber leicht ändern, wenn Ihnen 2-Ton-Sprünge zu einfach erscheinen! (Letzter Wert in Zeile 490 bestimmt die Sprungweite).
4. Haben Sie »2« gedrückt, so können Sie sich in den Intervall-Sprüngen trainieren! (c-c = Prime, c-d = Sekunde, c-e = Terz, c-f = Quarte, c-g = Quinte, c-a = Sexte und c-h = Septime). Geben Sie an, wieviele Intervalle Sie hören wollen. Nach jedem Intervall müssen Sie den Intervallnamen eintippen. (Erster Ton ist immer C!)
5. Haben Sie die »3« gewählt, so ist das Programm beendet und Ihr Gehör hoffentlich frisch gestärkt.

Torsten Niemietz)

```
1 REM MUSIKLEHRERTN1983
10 DIM Y$(50)
20 RANDOMIZE
30 A$="0C0A0808387838"
40 B$="1C1E1C1010101"
50 FOR A=1 TO 7
60 READ B,G$,H$
70 N(A)=B
80 H$(A)=G$
90 I$(A)=H$
100 NEXT A
110 CALL CLEAR
120 CALL SCREEN(2)
130 PRINT "MUSIK- LEHRER":"===
    ==========": : :"VON":" TO
    RSTEN NIEMIETZ":" HARBACHE
    R WEG 39":" D- 2800 BREMEN
     1": : :
140 PRINT : :"DRUECKE:":" #1))
     NOTENDIKTAT":" #2)) INTER
    VALL- TRAINING    #3)) END
    E": :
150 CALL HCHAR(1,1,96,32)
160 CALL VCHAR(2,1,97,22)
170 CALL VCHAR(2,32,97,22)
180 CALL HCHAR(24,1,96,32)
190 CALL SCREEN(16)
200 CALL CHAR(96,A$)
210 CALL CHAR(97,B$)
220 X=INT(RND*5)+1
230 CALL SOUND(-500,N(X),0,N(X
    +2),0)
240 CALL KEY(0,K,S)
250 CALL CHAR(96,B$)
260 CALL CHAR(97,A$)
270 IF S=0 THEN 200
280 IF (K<49)+(K>51)THEN 200
290 CALL CLEAR
300 CALL SOUND(500,294,5,392,0
    ,494,5)
310 CALL SOUND(500,349,0,392,0
    ,494,0)
320 CALL SOUND(1000,330,0,392,
    0,524,0)
330 ON K-48 GOTO 340,580,810
340 PRINT "NOTENDIKTAT": : :
350 INPUT "WIEVIELE TOENE? ":T
360 P=1
370 F=0
380 FOR Z=1 TO T
390 PRINT : :STR$(Z);". TON":
    :
400 CALL SOUND(-4000,N(P),0)
410 INPUT "TON=? ":S$
420 IF S$=H$(P)THEN 460
430 PRINT "FALSCH! RICHTIG WAE
    RE:";H$(P): :
440 F=F+1
450 GOTO 470
460 PRINT "RICHTIG!!": :
470 CALL SOUND(-300,44733,30)
480 INPUT "BITTE ^ENTER^ DRUEC
    KEN ...":S$
490 J=INT(RND*5)-2
500 IF (P+J<1)+(P+J>7)THEN 490
510 P=P+J
520 NEXT Z
530 CALL CLEAR
540 PRINT "AUSWERTUNG:": :"VON
    ";T;"TOENEN WAREN":"INSGES
    AMT";F;"FALSCH !": : : :
550 CALL SOUND(2000,262,0,330,
    5,392,5)
560 INPUT "BITTE ^ENTER^ DRUEC
    KEN ...":S$
570 GOTO 110
580 PRINT "INTERVALL- TRAINING
    ": : :"WIEVIELE INTERVALLE
    ";
590 INPUT T
600 F=0
610 FOR Z=1 TO T
620 PRINT "ACHTUNG !!!": :
630 CALL SOUND(1000,262,0)
640 J=INT(RND*7)+1
650 CALL SOUND(1000,N(J),0)
660 INPUT "INTERVALL =?":S$
670 IF S$=I$(J)THEN 720
680 CALL SOUND(200,220,0)
690 PRINT "FALSCH! RICHTIG GEW
    ESEN":"WAERE ";I$(J): : :
700 F=F+1
710 GOTO 740
720 CALL SOUND(150,1400,5)
730 PRINT "RICHTIG!!": : :
740 NEXT Z
750 INPUT "BITTE ^ENTER^ DRUEC
    KEN ...":S$
760 CALL CLEAR
770 PRINT "AUSWERTUNG:": :"VON
    ";T;"INTERVALLEN WAREN":"I
    NSGESAMT";F;"FALSCH!": : :
780 CALL SOUND(2000,262,0,330,
    5,392,5)
790 INPUT "BITTE ^ENTER^ DRUEC
    KEN ...":S$
800 GOTO 110
810 DATA 262,C,PRIME,294,D,SEK
    UNDE,330,E,TERZ,349,F,QUAR
    TE,392,G,QUINTE,440,A,SEXT
    E,494,H,SEPTIME
```

*Listing »Musiklehrer« zum Einüben der Tonleiter*

# Tiefe Töne hochgerechnet

**Möchten Sie sich Baßreflex-Lautsprecherboxen selbst bauen? Hierbei sind, neben den Eigenschaften des Lautsprechers, die Länge des Resonanzrohres sowie die idealen Abmessungen des Gehäuses für die Klangqualität von entscheidender Bedeutung. Für die Länge des Resonanzrohres wird Ihnen der TI 99/4A (in TI-Basic), für die Gehäuseabmessungen der Spectrum (16 KByte), die optimalen Werte berechnen.**

Da wohl niemand die Zeit hat, die beste Lösung für eine gute Baßbox auf experimentellem Wege zu finden, sind diese beiden Berechnungsprogramme eine prima Alternative, um Ihnen diese Arbeit abzunehmen. Dabei geht jedes der Programme von einer anderen Vorgabe aus. Das Programm »Baßreflex-Berechnung« dient zum Berechnen einer Baßreflex-Öffnung und setzt die Gehäusemaße als gegeben voraus.

Die Länge eines Baßreflex-Rohres ist nur mit einer etwas komplizierten Formel zu errechnen. Da man aber selten Unterlagen (auch nicht in Fachzeitschriften) dazu findet, soll hier das Rechenproblem dem TI 99/4A übertragen werden. Das Programm ist in TI-Basic geschrieben und läuft ohne jede Erweiterung auf dem Grundgerät. Es berücksichtigt automatisch die Dämmstoffe (zirka 10 Prozent des Gehäusevolumens), so daß es sehr einfach zu bedienen ist. Folgende Eingaben müssen gemacht werden:

☐ Gehäuse-Innenabmessungen in cm (Breite, Höhe, Tiefe, ohne Dämmstoff),

☐ Form der Baßreflex-Öffnung (rund oder eckig),

☐ Breite und Höhe, wenn die Öffnung eckig ist oder

☐ Durchmesser, wenn sie rund ist,

☐ Resonanzfrequenz des Lautsprecher-Chassis in Hertz.

Für die Größe der Baßreflex-Öffnung gilt als Faustregel: sie sollte gleich oder kleiner als die Öffnung des Baßlautsprechers sein.

(Gerhard Külkens)

```
80 !-------------------------+
   + BASSREFLEX-BERECHNUNG +
   +-------------------------+
90 !-------------------------+
   +   TI 99/4  TI-BASIC    +
   + COPY GERHARD KUELKENS +
   + VERSION 1,1 SEPT.1983 +
   +-------------------------+
100 CALL CLEAR
105 CALL SCREEN(15)
110 PRINT "** BASSREFLEX-BERECH
    NUNG **"
120 PRINT
125 PRINT "ALLE MASSE IN (cM)."
126 PRINT
130 PRINT "BITTE GEBEN SIE DURC
    H KOMMA-TA GETRENNT DIE INN
    EMMASSE EIN."
140 PRINT
150 INPUT "H,B,T, (cM) ":H1,B1,
    T1
160 PRINT
170 V=H1*B1*T1
180 PRINT V/1000;"LITER"
190 PRINT
200 PRINT "BASSREFLEX-OEFFNUNG
    ECKIG   ODER RUND?"
210 PRINT
220 INPUT "ECKIG=1-RUND=0 ? ":0
230 IF 0=1 THEN 240
240 IF 0=0 THEN 330
250 PRINT
260 PRINT "ECKIGE BASS-REFLEXOE
    FFNUNG!"
270 PRINT
280 PRINT "BITTE ABMESSUNGEN EI
    NGEBEN"
290 PRINT
300 INPUT "H,B (cM) ":H2,B2
310 A=H2*B2
320 GOTO 400
330 PRINT
340 PRINT "RUNDE BASS-REFLEXOEF
    FNUNG!"
350 PRINT
360 PRINT "BITTE DURCHMESSER EI
    NGEBEN"
370 PRINT
380 INPUT "cM ":D
390 A=D*D*0.785
400 PRINT
410 PRINT "GEBEN SIE DIE RESONA
    NZFRE-  QUENZ DES LAUTSPREC
    HERS EIN"
420 PRINT
430 INPUT "Hz ":F
440 F1=F*F
450 PI=3.1415927
460 REM BERECHNUNG
470 V=(V/100*90)
480 X=(34000*34000*A)/(39.5*F1*
    V)
490 Y=(X-((SQR(PI*A))*0.5))/2
500 Z=INT(Y*100+0.5)/100
510 CALL CLEAR
520 PRINT "-------- ERGEBNIS --
    -------"
530 PRINT
540 PRINT
550 PRINT "IHRE EINGABEN WAREN:
    "
560 PRINT
570 PRINT "INNEN-ABMESSUNGEN DE
    R BOX"
580 PRINT
590 PRINT "H=";H1;" B=";B1;" T=
    ";T1;" (cM)"
600 PRINT
610 IF 0=1 THEN 630
620 IF 0=0 THEN 670
630 PRINT "ECKIGE BASS-REFLEXOE
    FFNUNG"
640 PRINT
650 PRINT "H=";H2;" B=";B2;" (c
    M)"
660 GOTO 700
670 PRINT "RUNDE BASS-REFLEXOEF
    FNUNG"
680 PRINT
690 PRINT "D=";D;" (cM)"
700 PRINT
710 PRINT "RESONANZFREQUENZ D.
    CHASSIS"
720 PRINT
730 PRINT "F=";F;" (Hz)"
740 PRINT
750 PRINT
760 PRINT "LAENGE D. BASSREFLEX
    TUNNELS"
770 PRINT
780 PRINT "L=";Z;" (cM)"
790 PRINT
800 PRINT "NEUE BRECHNUNG?
    J/N "
810 CALL KEY(2,K,S)
820 IF S=0 THEN 810
830 IF K=2 THEN 100
840 IF K=15 THEN 850
850 END
860 !DIESES PROGRAMM DIENT
     ZUR BERECHNUNG DES BASS-
     REFEX-TUNNELS VON SELBST-
     BAU LAUTSPRECHER-BOXEN.
870 !ES ARBEITET IM DIALOG-
     BETRIEB.DER BENUTZER
     GIBT NUR DIE GEWUENSCH-
     TEN MASSE AN UND ERHAELT
     DIE LEANGE DES TUNNELS.
880 !HIER EINIGE MASSE ZUR
     UEBERPRUEFUNG OB ALLES
     RICHTIG EINGETIPPT
     WORDEN IST.
890 !HOEHE=50,BREITE=40
     TIEFE=30 cM.
900 !RESONANZFERQUENZ=60 Hz
910 !VOLUMEN=60 LITER
920 !ECKIGE OEFFNUNG:
     HOEHE=10,BREITE=40 cM
930 !ERGEBNIS: 21.25 cM
940 !RUNDE OEFFNUNG:
     DURCHMESSER=25 cM
950 !ERGEBNIS: 27.12 cM
```

**Beispiel:**

```
f0 = 70 Hz
V = 120 Liter
l = 10 cm   <--Kanallaenge
d = 14 cm   <--K-Durchmesser
Hoehe = 79.969393 cm
Breite = 49.965457 cm
Tiefe = 29.989139 cm
```

*Listing zur Berechnung von Baßreflexboxen mit dem TI 99/4A*

```
        BASSREFLEXBOXEN
      UND IHRE BERECHNUNG:

Folgende Groessen sind fuer die
Berechnung von Bassreflexboxen
von Bedeutung:

1. Die Resonanzfrequenz (F0[Hz])
   des Lautsprechers

2. Der Reflexkanaldurchmesser
   (d[cm])

3. Die Reflexkanallaenge (l[cm])


4. Das Gehaeusevolumen(V[Liter])

Setzen Sie die gesuchte Groesse
zu Null und geben Sie fuer die
anderen die entsp. Werte ein
```

```
  10 REM **Bassreflexboxen**
  20 REM **und ihre Berechnung**
  50 LET c=33000
  60 PRINT AT 0,7;"BASSREFLEXBOX
EN": PRINT AT 2,5;"UND IHRE BERE
CHNUNG:": PRINT AT 4,0;"Folgende
 Groessen sind fuer die Berechnu
ng von Bassreflexboxen  von Bede
utung:"
  65 PRINT AT 8,0;"1. Die Resona
nzfrequenz (F0[Hz])   des Lautsp
rechers"
  66 PRINT AT 11,0;"2. Der Refle
xkanaldurchmesser      (d[cm])"
  67 PRINT AT 14,0;"3. Die Refle
xkanallaenge (l[cm])"
  68 PRINT AT 17,0;"4. Das Gehae
usevolumen(V[Liter])"
  69 PRINT AT 19,0;"Setzen Sie d
ie gesuchte Groesse zu "; FLASH
1;"Null"; FLASH 0;" und geben Si
e fuer die  anderen die entsp. W
erte ein"
  70 PAUSE 300
 100 INPUT "Resonanzfreq.[Hz]=?"
,f0
 110 INPUT "R-Kanaldurchmesser[c
m]=?",d
 120 INPUT "R-Kanallaenge[cm]=?"
,l
 130 INPUT "Gehaeusevolumen[Lite
r]=?",v
 135 LET v1=v*1000
 140 IF f0=0 THEN LET f0=(c*d)/(
2*(SQR (PI*v1*(8*l+PI*d))))
 145 IF l=0 THEN LET l=((c↑2*d↑2
)/(4*PI*f0↑2*v1)-PI*d)*1/8
 150 IF d=0 THEN LET d=2*f0/c↑2*
(PI↑2*f0*v1+SQR (PI↑4*f0↑2*v1↑2+
8*PI*c↑2*v1*l))
 155 IF v=0 THEN LET v=(c↑2*d↑2/
(4*PI*f0↑2*(8*l+PI*d)))/1000
 156 CLS
 157 LET h=10*(1.6213*v↑(1/3))
 158 LET b=10*(1.013*v↑(1/3))
 159 LET t=10*(0.608*v↑(1/3))
 160 PRINT AT 0,0;"f0 = ";f0;" H
z"
 170 PRINT AT 2,0;"V = ";v;" Lit
er"
 175 LET l=INT (l*100): LET l=l/
100
 180 PRINT AT 4,0;"l = ";l;" cm"
;"   <--Kanallaenge"
 185 LET d=INT (d*100): LET d=d/
100
 190 PRINT AT 6,0;"d = ";d;" cm"
;"   <--K-Durchmesser"
 200 PRINT AT 8,0;"Hoehe = ";h;"
cm"
 210 PRINT AT 10,0;"Breite = ";b
;" cm"
 220 PRINT AT 12,0;"Tiefe = ";t;
" cm"
```

*Berechnungsbeispiel und Listing für den Spectrum*

Beim Programm für den Spectrum ist die Bedienung anders. Es sind vier Eingaben möglich:

- ☐ Resonanzfrequenz,
- ☐ Reflexkanaldurchmesser,
- ☐ Reflexkanallänge oder
- ☐ Volumen des Gehäuses.

Je nachdem, welche dieser Größen errechnet werden soll, muß die gesuchte Größe mit dem Wert »0« eingegeben werden. In den Programmzeilen 140 bis 155 wird vom Computer festgestellt, welche Eingabevariable Null ist. Er weiß dann, welchen Wert der Anwender wünscht. Jeweils im zweiten Teil dieser Zeilen wird eine der folgenden Formeln für die Rechnung benützt:

Resonanzfrequenzberechnung:

$$f_0 = \frac{c \cdot d}{2\sqrt{\pi \cdot V\,(8L + \pi \cdot d)}}$$

Gehäusevolumen:

$$V = \frac{c^2 \cdot d^2}{4\pi\, f_0^2\,(8L + \pi \cdot d)}$$

Baßreflexkanal:

$$L = \frac{1}{8}\left(\frac{c^2 \cdot d^2}{4\pi\, f_0^2 \cdot V} - \pi \cdot d\right)$$

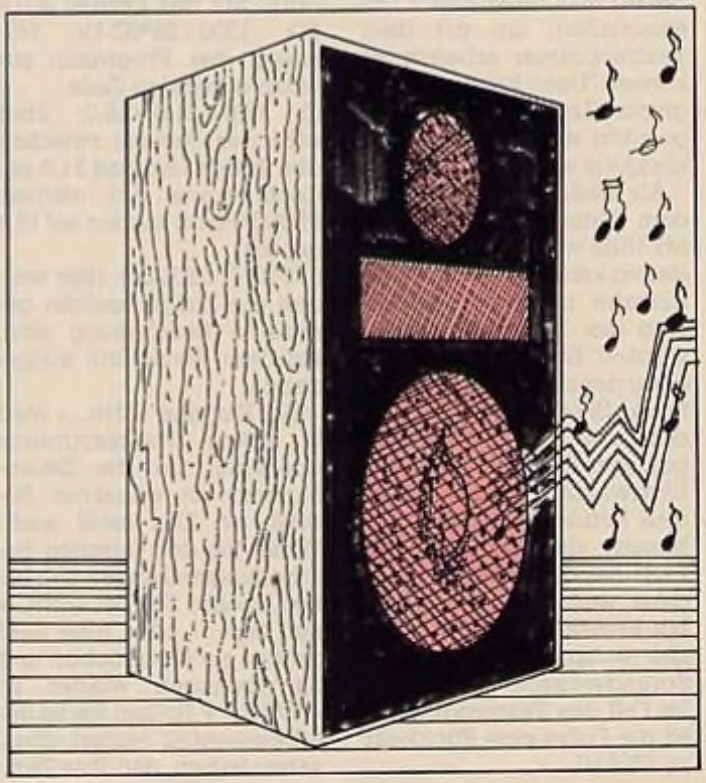
Durchmesser der Reflexkanalöffnung:

$$d = \frac{2 f_0}{c^2}\left(\pi^2 f_0\, V + \sqrt{\pi^4 f_0^2\, V^2 + 8\pi c^2 V L}\right)$$

dabei bedeutet:
V: Gehäusevolumen (cm³)
c: Schallgeschwindigkeit (33000 cm/s)
d: Reflexkanalöffnung (Durchmesser in cm)
$f_0$: Resonanzfrequenz des Lautsprechers im nicht eingebauten Zustand
x: Reflexkanallänge (cm)

Das Verhältnis von Höhe : Breite : Tiefe ist nach Klinger gewählt worden, also: 1,6213 : 1,013 : 0,608.

(Hermann Peters)



# Neues Futter für TI 99/4A

***Was lange währt wird gut, behauptet ein altes Sprichwort. Besitzer des TI 99/4A dürften bisher am Wahrheitsgehalt gezweifelt haben. Ohne vernünftige Literatur konnten sie nur nach der Methode Versuch und Irrtum in die tieferen Geheimnisse des TI 99/4A eindringen. Nun plötzlich erscheint eine ganze Flut an guten Büchern.***

Nicht nur, daß Texas Instruments höchst persönlich Bücher zum Thema herausbringt (wir haben bereits darüber berichtet), auch verschiedene Verlage und jetzt sogar Data Becker entdecken das Thema TI 99/4A. »TI-99, Tips & Tricks« nennt sich Data Beckers neues Buch und behauptet, eine »Fundgrube für den TI-99-Anwender« zu sein. Der Untertitel verspricht tatsächlich nicht zuviel. Zwar ist das Buch kein Nachschlagewerk für den Freak, für den normalen Anwender aber, der seine ersten Schritte mit dem Standardhandbuch bereits hinter sich hat, stellt es genau die richtige Fortsetzungslektüre dar. Vor allem die Mischung aus einfachen grundlegenden Erläuterungen zu den nicht so oft benutzten Befehlen (zum Beispiel CALL CHARPAT, CALL COINC, CALL ERR, CALL GCHAR etc.), echten Tips (unter anderem: Passwortroutine, Programmschutz, Speicherplatzeinsparungen und Beschleunigung der Kassettenaufzeichnung bei Datenspeicherung) sowie Beispielprogrammen für häufig benötigte Routinen (Wochentagsberechnung, Sortieren, Textverarbeitung) ist kurzweilig und praktisch. Obwohl vieles in diesem Buch von der erweiterten Basic-Version, dem Extended Basic, handelt, ist es auch und gerade für den Besitzer der schlichten Konsole geeignet. Die Erkenntnis am Schluß der Lektüre wird zwar darin bestehen, daß man unbedingt das Extended Basic-Modul braucht, aber diese Erkenntnis wird jeder als berechtigt bestätigen können, der das Modul bereits längere Zeit besitzt.

Ganz besonders lobenswert ist ein Kapitel gleich zu Beginn des Buches, in dem wichtige Befehle des weitverbreiteten Microsoft Basic mit dem TI-Basic verglichen werden. Für einige sind sogar in TI-Basic Ersatzroutinen angegeben. Dieses Kapitel dürfte endlich das Dilemma vieler TI-Besitzer lösen, die gute Programme umstricken wollen, aber mangels Möglichkeit zum Kennenlernen anderer Basic-Dialekte nur ihr TI-Basic kennen.

Der Autor, Karl Schwinn, wandelt auf dem schwierigen und schmalen Grat zwischen banaler Anfängerinformation und unverständlicher Insider-message. Seine Gratwanderung ist gut gelungen. Das Buch stellt übrigens inhaltlich keine Konkurrenz für bisherige Veröffentlichungen zum TI 99/4A dar, sondern ergänzt diese. Bedeutsame Fehler konnten keine entdeckt werden. Zum Preis von 49 Mark erhält der Anwender 300 Seiten gute Information im Gewand eines Paperback. (lg)

Info: Data Becker ISBN 3-89011-006-1, Merowingerstr. 30, 4000 Düsseldorf

# Paperware für Sharp-Benutzer

***Die Auswahl an Literatur für Sharp-Benutzer ist nicht gerade reichhaltig. Wir haben deswegen hier einmal zusammengefaßt, was es an neuerer Literatur gibt — auch wenn sicher nicht jedes Buch jeden Benutzer eines Sharp-Gerätes interessiert.***

»Programmierhandbuch für Sharp« nennt sich das 189seitige Buch von Burger/Hofacker/Schock/Schupp/Leistner, das im Hofacker-Verlag erschienen ist (49 Mark). Es handelt sich dabei um eine Sammlung von 59 Programmen, die zur Hälfte für den MZ80B bestimmt sind. Der Rest verteilt sich auf Programme für 80K und A (die auch auf dem Modell B laufen), für PC-1500 und PC-1211/12 sowie auf einige Programme, bei denen kein bestimmter Rechnertyp (er wurde wohl vergessen) angegeben ist. Wer nicht den MZ80B besitzt, muß das Buch genau durchblättern, um festzustellen, was sich für ihn eignet — aus dem Inhaltsverzeichnis ist es nicht einwandfrei zu ersehen. Bei der Sammlung handelt es sich um eine kunterbunte Mischung (knapp zur Hälfte Spiele), die vom simplen Neunzeiler »Errechnung der verbrauchten Liter auf 100 km« bis zum 19seitigen »Raumschiff Enterprise« reicht.

Betriebssystem-Listings für MZ80K, A und B, MZ1200 und 2000 (Basic, Assembler, FDOS und zum Teil Pascal), MZ3500 (EOS) und MZ700 (Basic) gibt es von Wolfgang Dommach (Köln). Eine »Sharp-Zeitung«, die MZ80 und 700 behandelt, gibt jetzt Fischel (Berlin) heraus; dort soll im Februar auch ein Buch über Software und Hardwareerweiterungen für den MZ80K erscheinen; bei der Firma gibt's außerdem unter der Bezeichnung »MZ80B intern« Programmiertips und Routinen zu Preisen zwischen 3 und 10.

Ein »Lehr- und Übungsbuch für den PC-1500« von Claus Peter Ortlieb ist im Verlag Vieweg erschienen (150 Seiten, 29,80 Mark). Es wendet sich vor allem an Benutzer ohne Programmierkenntnisse, ist sauber gemacht und führt in die Benutzung des Rechners sowie die Programmierung in Basic ein. Zwei Dutzend Programmbeispiele vor allem aus dem naturwissenschaftlich/technischen Bereich und Hinweise zum Umgang mit dem Zusatz CE 150 runden das ganze ab. Weitere Programmbeispiele für diesen Rechner und den PC-1251 finden sich in Dieter Langes Buch »Analyse elektrischer und elektronischer Netzwerke mit Basic-Programmen« (Verlag Vieweg, 234 Seiten, 29,80 Mark).

Für die Taschencomputer PC-1210/11/12 gab es bisher schon ein »Lehr- und Übungsbuch« von Horst Kreht. Im gleichen Verlag (Vieweg) erschien jetzt eine 75seitige Sammlung »Basic-Programme für den PC-1211/12« von Harald Schumny. Für 19,80 Mark gibt es damit eine recht eigenwillige Mischung von zehn Programmen für Ballonnavigation, Lotto- und Primzahlengenerierung, Fakultätsberechnung, Skatabrechnung, Erstellen von Bundesligatabellen, Einsatz des PC als druckender Tischrechner sowie Programme aus der Chemie und Atomtheorie.

# Funktionsplotter für TI 99/4A

**Das Programm ermöglicht es, in Verbindung mit dem Modul Extended Basic und dem direkt anschließbaren TI-Thermodrucker, bis zu neun Funktionen gleichzeitig zu plotten.**

Die Auflösung läßt sich in X-Richtung durch Vorgabe der Schrittweite h beliebig wählen. In Y-Richtung (quer zur Papiertransportrichtung) besteht eine Beschränkung auf 96 Punkte, die aber bei entsprechender Bereichswahl durch Nebeneinanderreihen der Druckerstreifen umgangen werden kann. Diese Einschränkung resultiert aus der Tatsache, daß zwischen den Zeichen in dieser Richtung jeweils ein Abstand in Punktbreite besteht. Um dennoch eine gleichmäßige Auflösung zu erhalten, wird nur jeder zweite Punkt eines Zeichens angesteuert, also je drei pro Zeichen (die beiden äußeren und der mittlere, 3 x 32 = 96). Es lassen sich aber die Punkte nicht einzeln schalten, sondern nur ganze Zeichen. Deshalb muß das Plotten den Umweg über die Zeichendefinition nehmen, was leider eine umständliche Programmierung erfordert. Wenn nötig zeichnet das Programm die X- und Y-Achsen automatisch.

## Funktionen stehen in den Zeilen 140 bis 220

Die Zeilen 140 bis 220 beinhalten die Funktionen. Für unbenutzte Zeilen wird eine Null eingesetzt. ON WARNING NEXT in Zeile 230 unterdrückt die Warnung NUMERIC OVERFLOW, falls Funktionen Unstetigkeitsstellen und ähnliches aufweisen. In den Zeilen 240 bis 380 werden die Bereichswerte des Ausdrucks eingegeben. Danach kann im Falle irrtümlicher Eingabe die Eingabewiederholung durch Taste »N« erzwungen werden. Bleiben Werte unverändert, braucht bloß die ENTER-Taste gedrückt werden. Mit C wird der Programmablauf fortgesetzt, mit E beendet.

## Thermodrucker benützt Video-RAM

Bei Beginn des Ausdrucks leuchtet der Bildschirm hellgrün. Zeile 320 überprüft die Eingabewerte. Sollten Werte nicht plausibel sein, wird eine neue Eingabe verlangt. Die Zeilen 390 bis 420 bringen die Bereichswerte zum Ausdruck. In Zeile 440 wird die Position der X-Achse berechnet.

Der Thermodrucker besitzt den Vorteil, daß man ihn vom eigenen festen Zeichensatz auf den variablen des Video-Display-Prozessor-RAMs umschalten kann. In Zeile 450 geschieht das durch die Option U in der Gerätebezeichnung sowie die Unterdrückung des Zeilenabstandes durch S. In den Zeilen 460 bis 480 wird das erste Byte des Zeichen-

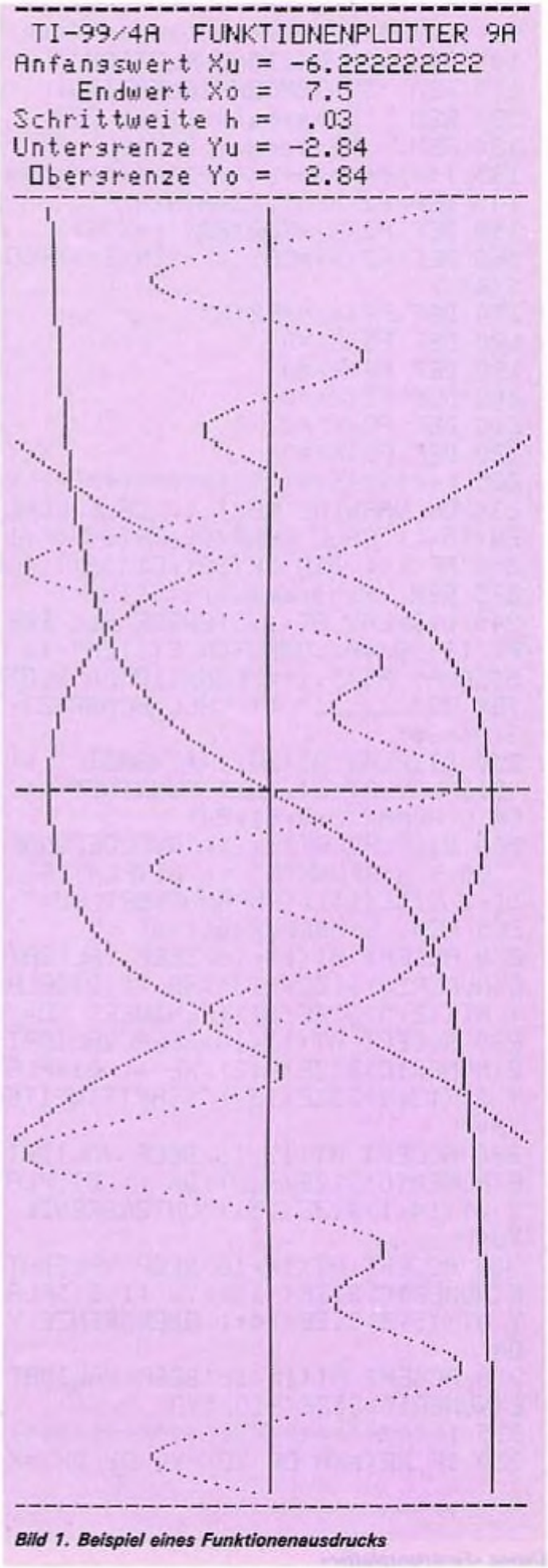



*Bild 1. Beispiel eines Funktionenausdrucks*

```
95 !*****************************
100 REM  "FUNKTIONENPLOTTER"
110 REM  SYSTEMCONFIGURATION:
120 REM   >Thermaldrucker 32Z
130 REM   >Extended BASIC
135 !*****************************
140 DEF F1(X)=1.72*ATN(X)
150 DEF F2(X)=SQR(ABS(6-X^2))
160 DEF F3(X)=COS(X)-SIN(2*X)+CO
S(4*X)
170 DEF F4(X)=-F2(X)
180 DEF F5(X)=0
190 DEF F6(X)=0
200 DEF F7(X)=0
210 DEF F8(X)=0
220 DEF F9(X)=0
225 !*****************************
230 ON WARNING NEXT :: CALL SCRE
EN(15):: CALL CHAR(96,RPT$("0",1
2)&"FF"):: DIM SK(32),C$(32)
235 REM  Programmvorstellung:
240 DISPLAY AT(1,3)ERASE ALL BEE
P:"TI-99/4A COMPUTER SYSTEM" ::
DISPLAY AT(5,1):"FUNKTIONEN PLOT
TER 32Z_____" :: CALL HCHAR(23,
3,96,28)
250 DISPLAY AT(24,1):"XBASIC" ::
 DISPLAY AT(24,22):"RUNNING" ::
CALL HCHAR(2,5,61,24)
260 DISPLAY AT(8,1):"AUFLOESUNG:
  96 * X  PUNKTE" :: DISPLAY AT(
11,1)SIZE(15):"ANFANGSWERT XU="
265 REM  Eingaberoutine:
270 ACCEPT AT(11,16)BEEP VALIDAT
E(NUMERIC)SIZE(-13):XA :: DISPLA
Y AT(12,5)SIZE(11):"ENDWERT XO="
280 ACCEPT AT(12,16)BEEP VALIDAT
E(NUMERIC)SIZE(-13):XE :: DISPLA
Y AT(13,1)SIZE(15):"SCHRITTWEITE
 H="
290 ACCEPT AT(13,16)BEEP VALIDAT
E(NUMERIC)SIZE(-13):DX :: DISPLA
Y AT(14,1)SIZE(15):"UNTERGRENZE
YU="
300 ACCEPT AT(14,16)BEEP VALIDAT
E(NUMERIC)SIZE(-13):YU :: DISPLA
Y AT(15,2)SIZE(14):"OBERGRENZE Y
O="
310 ACCEPT AT(15,16)BEEP VALIDAT
E(NUMERIC)SIZE(-13):YO
315 !*****************************
320 IF XE<=XA OR YO<=YU OR DX>=X
E-XA THEN DISPLAY AT(18,1)BEEP:"
* UNERLAUBTE EINGABE!" :: GOTO 2
60 !Eingabeueberpruefung
330 DISPLAY AT(18,1):"PRESS C FO
R CONTINUE" :: DISPLAY AT(19,1):
"PRESS N FOR NEW INPUT" :: DISPL
AY AT(20,1):"PRESS I FOR INTERRU
PT"
340 DISPLAY AT(21,1)BEEP:"PRESS
E FOR EXIT"
345 !*****************************
350 CALL KEY(0,K,S):: IF S=0 THE
N 350 !Tastaturabfrage
360 IF K=78 THEN 260
370 IF K=69 THEN CALL CLEAR :: E
ND
380 IF K<>67 THEN 350
385 !*****************************
386 REM Ausdruck der Eingabe
390 OPEN #1:"TP.E",OUTPUT
400 PRINT #1:RPT$(CHR$(28),32);"
 TI-99/4A  FUNKTIONENPLOTTER 9A"
:;:"Anfangswert Xu = ";XA:"    E
ndwert Xo = ";XE:"Schrittweite h
 = ";DX
410 PRINT #1:"Untergrenze Yu = "
;YU:" Obergrenze Yo = ";YO:RPT$(
"-",32)
420 CLOSE #1 :: X=XA
425 !*****************************
430 CALL SCREEN(4):: RESTORE ::
FOR I=0 TO 7 :: READ S$(I):: NEX
T I
440 IF SGN(YO)=SGN(YU)THEN W(0)=
0 ELSE W(0)=INT(1-95*YU/(YO-YU))
!Bestimmung der X-Achse
450 OPEN #1:"TP.E.U.S",OUTPUT
460 FOR I=1 TO 32
470 C$(I)="00"
480 NEXT I
485 !*****************************
486 REM   Beginn der Zeichen=
487 REM   definition:
490 FOR II=1 TO 7
500 W(1)=F1(X):: W(2)=F2(X):: W(
3)=F3(X):: W(4)=F4(X):: W(5)=F5(
X):: W(6)=F6(X):: W(7)=F7(X):: W
(8)=F8(X):: W(9)=F9(X)
510 FOR I=1 TO 9
520 W(I)=INT(95*(W(I)-YU)/(YO-YU
)+1)
530 IF W(I)=W(0)OR W(I)<1 OR W(I
)>96 THEN W(I)=0
```

*Listing »Funktionsplotter«*

```
540 NEXT I
550 FOR I=1 TO 8
560 FOR S=I TO 8
570 IF W(I)=W(S+1)THEN W(I)=0
580 NEXT S :: NEXT I
590 FOR I=1 TO 32 :: SK(I)=0 ::
NEXT I
600 FOR N=0 TO 9
610 IF W(N)=0 THEN 630
620 H=INT((W(N)-1)/3+1):: SK(H)=
SK(H)+2^(W(N)-3*(H-1)-1)
630 NEXT N
640 FOR I=1 TO 32
650 IF X=0 OR SGN(X+DX)-SGN(X)=2
 THEN C$(I)=C$(I)&"FF" ELSE C$(I
)=C$(I)&S$(SK(I))
660 NEXT I
670 X=X+DX :: CALL KEY(0,K,S)
680 IF X>XE OR K=73 THEN GOSUB 7
20 :: PRINT #1:RPT$("-",32):: CL
OSE #1 :: CALL SOUND(1000,110,4,
130,3):: CALL SCREEN(15):: GOTO
350
690 NEXT II
700 GOSUB 720 :: GOTO 460
705 !*****************************
710 DATA 00,40,10,50,04,44,14,54
715 REM  Unterprogramm fuer
716 REM  Ausdruck:
720 Z$=""
730 FOR I=1 TO 32 :: CALL CHAR(1
11+I,C$(I)):: Z$=Z$&CHR$(111+I):
: NEXT I
740 PRINT #1:Z$ :: RETURN
```

*Listing »Funktionsplotter« (Schluß)*

musters jedes der 32 Zeichen einer Ausgabezeile auf Null gesetzt, da der Drucker aufgrund seiner 5 x 7-Matrix nur auf die folgenden 7 zugreift. In der Zeile 500 werden die aktuellen Funktionswerte aller neun Funktionen berechnet. In den Zeilen 510 bis 540 werden sie auf Werte zwischen 1 und 96 entsprechend den 96 Druckpositionen umgerechnet. Werte außerhalb des Bereichs bekommen Null zugeordnet. Die Zeilen 550 bis 580 eliminieren gleiche Werte, weil bei der folgenden Stringauswahl nur jeder Wert einmal existieren darf. Den drei Druckpunktpositionen eines Zeichens entspricht je ein Wert ($2^0$, $2^1$, $2^2$), der einer zeichenabhängigen Variablen SK(H) hinzuaddiert wird. Bei mehrfachen gleichen Funktionswerten würde eine zusätzliche Addition und damit eine Auswahl falscher Punkte erfolgen. Die Zeilen 600 bis 630 berechnen nun den Wert entsprechend der in einem Zeichen angesprochenen Punkte. In Zeile 650 wird dann entsprechend dem Wert von SK(H) einer der acht möglichen in Zeile 710 stehenden Stringwerte ausgewählt und an den Zeichenstring C$(I) angefügt.

| | |
|---|---|
| X | unabhängige Variable der Funktionen |
| XA | X-Anfangswert |
| XE | X-Endwert |
| DX | Schrittweite von X |
| YU | Y-Untergrenze |
| YO | Y-Obergrenze |
| K | Tastencode einer gedrückten Taste |
| S | Tastaturstatusvariable |
| II,I,N,S | Laufvariable |
| W(0) | Position der X-Achse (1 bis 96 oder 0) |
| W(1-9) | Druckpositionen der neun Funktionswerte |
| S$(0-7) | spezifiziert die acht möglichen Punktemuster, erhält die Werte des DATA-Statements in Zeile 710. |
| SK(1-32) | numerischer Wert (0-7), nach dem S$ ausgewählt wird. |
| C$(1-32) | Zeichenmuster für jedes der 32 Zeichen einer Druckzeile |
| H | Hilfsvariable, enthält Zeichenposition |
| Z$ | 32-stelliger Ausgabestring |

*Variablen-Liste zum »Funktionsplotter«*

### *Die Funktion wird durch Umdefinition von Zeichen erzeugt*

Zeile 670 erhöht X um die Schrittweite h. Zeile 680 beendet den Ausdruck, falls der X-Endwert erreicht oder die Taste I gedrückt wurde, wobei der Bildschirm wieder eine graue Farbe annimmt. Andernfalls erfolgt der Ausdruck einer Zeile und die Berechnung beginnt erneut in Zeile 490. Die Subroutine in den Zeilen 720 bis 740 übernimmt den Zusammenbau des 32stelligen Ausgabestrings und den Ausdruck.

In Bild 1 auf Seite 85 ist ein Beispielausdruck von vier überlagerten Funktionen zu sehen. Diese Funktionen stehen im Listing in den Zeilen 140 bis 170 und müssen vor der Eingabe eigener Funktionen selbstverständlich gelöscht werden.

(Willibald Stronegger)

# Morsen lernen — schnell und einfach

TI 99/4A

***Wer sein Radio einmal auf Kurzwelle einstellt und ein wenig in den Bändern sucht, kann sie noch hören — die Morsezeichen. Will jemand eine A- oder B-Lizenz als Funkamateur erlangen, muß er sie sogar erlernen. Mit dem Computer als Lehrer geht es leichter.***

Wer würde sich nicht an den einen oder anderen alten Abenteuerfilm erinnern, in dem ein Funker in dramatischen Momenten seine Morsetaste zurechtrückte und sein »SOS« in den Äther schickte. Kaum waren die ersten Morsezeichen als Antwort im Empfänger zu hören, konnte das Happy-End als gesichert gelten.

Im Prinzip waren diese Darstellungen richtig. Besonders die See- und Luftfahrt hätten in ihrer Geschichte unzählige Opfer mehr zu beklagen ohne die Erfindung Samuel Morses. Morsefunk ist die Funktechnik, die mit dem geringsten technischen Aufwand die höchste Effizienz erreicht, und auch heute werden in den Ländern der dritten Welt die meisten Funkverbindungen auf diese Weise abgewickelt. Sogar Hochseekreuzer mit modernsten Telex- und Sprechfunkstationen an Bord müssen laut Gesetz nach wie vor eine Morsefunkanlage für den Notfall besitzen.

So scheint es durchaus sinnvoll, wenn angehende Funkamateure, die den Sendebetrieb auf Kurzwelle durchführen wollen, immer noch Morsen lernen müssen. Mit dem hier vorgestellten Programm (Listing 1) kann man das Hören und Verstehen von Morsezeichen üben. Man bezeichnet das als »Aufnehmen«. Das Programm ist damit nicht nur für angehende Funkamateure interessant, sondern auch für diejenigen, die gerne mal in die Kurzwellenbänder reinhören und vom Inhalt etwas verstehen möchten.

```
100 REM  ***MORSE-TUTOR***
110 DIM A$(46)
120 DIM F$(45)
130 GOSUB 690
140 CALL CLEAR
150 PRINT "MORSE-UEBUNGSPROGRAMM
"
160 PRINT ::
170 INPUT "MANUELL ODER AUTOMATI
K? M/A  ":L$
180 IF L$="M" THEN 260
190 INPUT "WELCHE LEKTION WUENSC
HEN SIE? 1-19  ":R
200 ON R GOSUB 790,1010,1220,143
0,1650,1870,2090,2310,2530,2750,
2970,3250,3410,3630,3850,4070,42
90,4510,4730
210 CALL CLEAR
220 GOTO 270
230 INPUT "NOCHEINMAL? J/N   ":J
N$
240 IF JN$="J" THEN 170
250 GOTO 230
260 INPUT "MESSAGE?   ":TEXT$
270 PRINT TEXT$
280 L=LEN(TEXT$)
290 FOR I=1 TO L
300 C=ASC(SEG$(TEXT$,I,L))
310 IF C=32 THEN 410
320 IF C=40 THEN 430
330 IF C=41 THEN 430
340 IF C>43 THEN 350
350 IF C<59 THEN 450
360 IF C=61 THEN 470
370 IF C=63 THEN 490
380 IF C>64 THEN 390
390 IF C<91 THEN 510
400 GOTO 520
410 C=C-31
420 GOTO 520
430 C=C-38
440 GOTO 520
450 C=C-40
460 GOTO 520
470 C=C-42
480 GOTO 520
490 C=C-43
500 GOTO 520
510 C=C-44
520 K$=A$(C)
530 L=LEN(K$)
540 FOR S=1 TO L
550 T$=SEG$(K$,S,1)
560 IF T$="1" THEN 650
570 IF T$="2" THEN 630
580 IF T$="3" THEN 670
590 NEXT S
600 NEXT I
610 CALL CLEAR
620 GOTO 150
630 CALL SOUND(10,1000,1)
640 GOTO 671
650 CALL SOUND(100,1000,1)
660 GOTO 671
670 CALL SOUND(150,1000,30)
671 CALL SOUND(1,1000,30)
680 GOTO 590
690 REM ***EINGABE-ROUTINE***

700 RESTORE 750
710 FOR LOOP=1 TO 46
720 READ ELEMENT$
730 A$(LOOP)=ELEMENT$
740 NEXT LOOP
750 DATA 3,12112,121121,112211,1
22221,212121,12212,11111,21111,2
2111,22211,22221,22222
760 DATA 12222,11222,11122,11112
,111222,12221,221122,21,1222,121
2,122,2,2212,112,2222
770 DATA 22,2111,121,2122,11,12,
111,2112,1121,212,222,1,221,2221
,122,1221,1211,1122
780 RETURN
790 REM  LEKTION 1: V E M K 2

800 CALL CLEAR
810 PRINT "LEKTION 1"
820 PRINT
830 PRINT " ZEICHEN :
        V E M K 2"
840 PRINT
850 RESTORE 900
860 FOR LOOP=1 TO 5
870 READ ELEMENT$
880 F$(LOOP)=ELEMENT$
890 NEXT LOOP
900 DATA V,E,M,K,2
910 RANDOMIZE
920 FOR I=1 TO 10
930 N$=N$&" "
940 FOR J=1 TO 5
950 V=INT(5*RND)+1
960 N$=N$&F$(V)
970 NEXT J
980 NEXT I
990 TEXT$=N$
1000 RETURN
1010 REM   LEKTION 2: H O 0 .
1020 CALL CLEAR
1030 PRINT "LEKTION 2"
1040 PRINT
```

*Listing Morseübungsprogramm*

```
1050 PRINT "ZEICHEN:
       H O 0 ."
1060 RESTORE 1110
1070 FOR LOOP=1 TO 4
1080 READ ELEMENT$
1090 F$(LOOP)=ELEMENT$
1100 NEXT LOOP
1110 DATA H,O,0,.
1120 RANDOMIZE
1130 FOR I=1 TO 10
1140 N$=N$&" "
1150 FOR J=1 TO 5
1160 V=INT(4*RND)+1
1170 N$=N$&F$(V)
1180 NEXT J
1190 NEXT I
1200 TEXT$=N$
1210 RETURN
1220 REM  LEKTION 3:(L1+L2)
1230 CALL CLEAR
1240 PRINT "LEKTION 3"
1250 PRINT
1260 PRINT "ZEICHEN:
       V E M K 2
   H O 0 ."
1270 RESTORE 1320
1280 FOR LOOP=1 TO 9
1290 READ ELEMENT$
1300 F$(LOOP)=ELEMENT$
1310 NEXT LOOP
1320 DATA V,E,M,K,2,H,O,0,.
1330 RANDOMIZE
1340 FOR I=1 TO 10
1350 N$=N$&" "
1360 FOR J=1 TO 5
1370 V=INT(9*RND)+1
1380 N$=N$&F$(V)
1390 NEXT J
1400 NEXT I
1410 TEXT$=N$
1420 RETURN
1430 REM  LEKTION 4: A R S 6 ,

1440 CALL CLEAR
1450 PRINT "LEKTION 4"
1460 PRINT
1470 PRINT "ZEICHEN:
       A R S 6 ,"
1480 PRINT
1490 RESTORE 1540
1500 FOR LOOP=1 TO 5
1510 READ ELEMENT$
1520 F$(LOOP)=ELEMENT$
1530 NEXT LOOP
1540 DATA A,R,S,6,,
1550 RANDOMIZE
1560 FOR I=1 TO 10
1570 N$=N$&" "
1580 FOR J=1 TO 5
1590 V=INT(5*RND)+1
1600 N$=N$&F$(V)
1610 NEXT J
1620 NEXT I
1630 TEXT$=N$
1640 RETURN
1650 REM  LEKTION 5:I N 8 ?
1660 CALL CLEAR
1670 PRINT "LEKTION 5"
1680 PRINT
1690 PRINT "ZEICHEN:
       I N 8 ?"
1700 PRINT
1710 RESTORE 1760
1720 FOR LOOP=1 TO 4
1730 READ ELEMENT$
1740 F$(LOOP)=ELEMENT$
1750 NEXT LOOP
1760 DATA I,N,8,?
1770 RANDOMIZE
1780 FOR I=1 TO 10
1790 N$=N$&" "
1800 FOR J=1 TO 5
1810 V=INT(4*RND)+1
1820 N$=N$&F$(V)
1830 NEXT J
1840 NEXT I
1850 TEXT$=N$
1860 RETURN
1870 REM  LEKTION 6:(L4+L5)
1880 CALL CLEAR
1890 PRINT "LEKTION 6"
1900 PRINT
1910 PRINT "ZEICHEN:
       A R S 6 ,
    I N 8 ?"
1920 PRINT
1930 RESTORE 1980
1940 FOR LOOP=1 TO 9
1950 READ ELEMENT$
1960 F$(LOOP)=ELEMENT$
1970 NEXT LOOP
1980 DATA A,R,S,6,,,I,N,8,?
1990 RANDOMIZE
2000 FOR I=1 TO 10
2010 N$=N$&" "
2020 FOR J=1 TO 5
2030 V=INT(9*RND)+1
2040 N$=N$&F$(V)
2050 NEXT J
2060 NEXT I
2070 TEXT$=N$
2080 RETURN
2090 REM  LEKTION 7:(L3+L6)
2100 CALL CLEAR
2110 PRINT "LEKTION 7"
2120 PRINT
2130 PRINT "ZEICHEN:V E M K 2 H
O 0 . A R S 6 , I N 8 ?"
2140 PRINT
2150 RESTORE 2200
2160 FOR LOOP=1 TO 18
2170 READ ELEMENT$
```

***Listing Morseübungsprogramm (Fortsetzung)***

```
2180 F$(LOOP)=ELEMENT$
2190 NEXT LOOP
2200 DATA V,E,M,K,2,H,O,0,.,A,R,
S,6,,,I,N,8,?
2210 RANDOMIZE
2220 FOR I=1 TO 10
2230 N$=N$&" "
2240 FOR J=1 TO 5
2250 V=INT(18*RND)+1
2260 N$=N$&F$(V)
2270 NEXT J
2280 NEXT I
2290 TEXT$=N$
2300 RETURN
2310 REM  LEKTION 8:C T Y 3 1

2320 CALL CLEAR
2330 PRINT "LEKTION 8"
2340 PRINT
2350 PRINT "ZEICHEN:
       C T Y 3 1"
2360 PRINT
2370 RESTORE 2420
2380 FOR LOOP=1 TO 5
2390 READ ELEMENT$
2400 F$(LOOP)=ELEMENT$
2410 NEXT LOOP
2420 DATA C,T,Y,3,1
2430 RANDOMIZE
2440 FOR I=1 TO 10
2450 N$=N$&" "
2460 FOR J=1 TO 5
2470 V=INT(5*RND)+1
2480 N$=N$&F$(V)
2490 NEXT J
2500 NEXT I
2510 TEXT$=N$
2520 RETURN
2530 REM  LEKTION 9:L Q ( )
2540 CALL CLEAR
2550 PRINT "LEKTION 8"
2560 PRINT
2570 PRINT "ZEICHEN:
       L Q ( )"
2580 PRINT
2590 RESTORE 2640
2600 FOR LOOP=1 TO 4
2610 READ ELEMENT$
2620 F$(LOOP)=ELEMENT$
2630 NEXT LOOP
2640 DATA L,Q,(,)
2650 RANDOMIZE
2660 FOR I=1 TO 10
2670 N$=N$&" "
2680 FOR J=1 TO 5
2690 V=INT(4*RND)+1
2700 N$=N$&F$(V)
2710 NEXT J
2720 NEXT I
2730 TEXT$=N$
2740 RETURN
2750 REM  LEKTION 10:(L8+L9)
2760 CALL CLEAR
2770 PRINT "LEKTION 10"
2780 PRINT
2790 PRINT "ZEICHEN:
       C T Y 3 1
    L Q ( )"
2800 PRINT
2810 RESTORE 2860
2820 FOR LOOP=1 TO 9
2830 READ ELEMENT$
2840 F$(LOOP)=ELEMENT$
2850 NEXT LOOP
2860 DATA C,T,Y,3,1,L,Q,(,)
2870 RANDOMIZE
2880 FOR J=1 TO 10
2890 N$=N$&" "
2900 FOR I=1 TO 5
2910 V=INT(9*RND)+1
2920 N$=N$&F$(V)
2930 NEXT I
2940 NEXT J
2950 TEXT$=N$
2960 RETURN
2970 REM   LEKTION 11:(L7+L10)
2980 CALL CLEAR
2990 PRINT "LEKTION 11"
3000 PRINT
3010 PRINT "ZEICHEN:
       V E M K 2 H O 0 .
   A R S 6 , I N 8 ?          C
 T Y 3 1 L Q ( )"
3020 PRINT
3030 RESTORE 3080
3040 FOR LOOP=1 TO 27
3050 READ ELEMENT$
3060 F$(LOOP)=ELEMENT$
3070 NEXT LOOP
3080 DATA V,E,M,K,2,H,O,0,.,A,R,
S,6,,,I,N,8,?,C,T,Y,3,1,L,Q,(,)
3090 RANDOMIZE
3100 FOR I=1 TO 10
3110 N$=N$&" "
3120 FOR J=1 TO 5
3130 V=INT(27*RND)+1
3140 N$=N$&F$(V)
3150 NEXT J
3160 NEXT I
3170 TEXT$=N$
3180 RETURN
3190 REM  LEKTION 12:Z D U 4 -

3200 CALL CLEAR
3210 PRINT "LEKTION 12"
3220 PRINT
3230 PRINT "ZEICHEN:
       Z D U 4 -"
3240 PRINT
3250 RESTORE 3300
3260 FOR LOOP=1 TO 5
3270 READ ELEMENT$
```

*Listing Morseübungsprogramm (Fortsetzung)*

```
3280 F$(LOOP)=ELEMENT$
3290 NEXT LOOP
3300 DATA Z,D,U,4,-
3310 RANDOMIZE
3320 FOR I=1 TO 10
3330 N$=N$&" "
3340 FOR J=1 TO 5
3350 V=INT(5*RND)+1
3360 N$=N$&F$(V)
3370 NEXT J
3380 NEXT I
3390 TEXT$=N$
3400 RETURN
3410 REM  LEKTION 13:G X 9 :
3420 CALL CLEAR
3430 PRINT "LEKTION 13"
3440 PRINT
3450 PRINT "ZEICHEN:
       G X 9 :"
3460 PRINT
3470 RESTORE 3520
3480 FOR LOOP=1 TO 4
3490 READ ELEMENT$
3500 F$(LOOP)=ELEMENT$
3510 NEXT LOOP
3520 DATA G,X,9,:
3530 RANDOMIZE
3540 FOR I=1 TO 10
3550 N$=N$&" "
3560 FOR J=1 TO 5
3570 V=INT(4*RND)+1
3580 N$=N$&F$(V)
3590 NEXT J
3600 NEXT I
3610 TEXT$=N$
3620 RETURN
3630 REM  LEKTION 14:(L12+L13)
3640 CALL CLEAR
3650 PRINT "LEKTION 14"
3660 PRINT
3670 PRINT "ZEICHEN:
       Z D U 4 -
    G X 9 :"
3680 PRINT
3690 RESTORE 3740
3700 FOR LOOP=1 TO 9
3710 READ ELEMENT$
3720 F$(LOOP)=ELEMENT$
3730 NEXT LOOP
3740 DATA Z,D,U,4,-,G,X,9,:
3750 RANDOMIZE
3760 FOR I=1 TO 10
3770 N$=N$&" "
3780 FOR J=1 TO 5
3790 V=INT(9*RND)+1
3800 N$=N$&F$(V)
3810 NEXT J
3820 NEXT I
3830 TEXT$=N$
3840 RETURN
3850 REM  LEKTION 15:(L11+L14)
3860 CALL CLEAR
3870 PRINT "LEKTION 15"
3880 PRINT
3890 PRINT "ZEICHEN:
      VEMK2HO0.ARS6,IN8?CTY31LQ
()ZDU4-GX9:"
3900 PRINT
3910 RESTORE 3960
3920 FOR LOOP=1 TO 36
3930 READ ELEMENT$
3940 F$(LOOP)=ELEMENT$
3950 NEXT LOOP
3960 DATA V,E,M,K,2,H,O,0,.,A,R,
S,6,,,I,N,8,?,C,T,Y,3,1,L,Q,(,),
Z,D,U,4,-,G,X,9,:
3970 RANDOMIZE
3980 FOR I=1 TO 10
3990 N$=N$&" "
4000 FOR J=1 TO 5
4010 V=INT(36*RND)+1
4020 N$=N$&F$(V)
4030 NEXT J
4040 NEXT I
4050 TEXT$=N$
4060 RETURN
4070 REM  LEKTION 16:B F W 7 =
4080 CALL CLEAR
4090 PRINT "LEKTION 16"
4100 PRINT
4110 PRINT "ZEICHEN:
       B F W 7 ="
4120 PRINT
4130 RESTORE 4180
4140 FOR LOOP=1 TO 5
4150 READ ELEMENT$
4160 F$(LOOP)=ELEMENT$
4170 NEXT LOOP
4180 DATA B,F,W,7,=
4190 RANDOMIZE
4200 FOR I=1 TO 10
4210 N$=N$&" "
4220 FOR J=1 TO 5
4230 V=INT(5*RND)+1
4240 N$=N$&F$(V)
4250 NEXT J
4260 NEXT I
4270 TEXT$=N$
4280 RETURN
4290 REM  LEKTION 17:J P 5 /
4300 CALL CLEAR
4310 PRINT "LEKTION 17"
4320 PRINT
4330 PRINT "ZEICHEN:
       J P 5 /"
4340 PRINT
4350 RESTORE 4400
4360 FOR LOOP=1 TO 4
4370 READ ELEMENT$
4380 F$(LOOP)=ELEMENT$
4390 NEXT LOOP
4400 DATA J,P,5,/
```

***Listing Morseübungsprogramm (Fortsetzung)***

```
4410 RANDOMIZE
4420 FOR I=1 TO 10
4430 N$=N$&" "
4440 FOR J=1 TO 5
4450 V=INT(4*RND)+1
4460 N$=N$&F$(V)
4470 NEXT J
4480 NEXT I
4490 TEXT$=N$
4500 RETURN
4510 REM  LEKTION 18
4520 CALL CLEAR
4530 PRINT "LEKTION 18"
4540 PRINT
4550 PRINT "ZEICHEN:
       B F W 7 =
    J P 5 /"
4560 PRINT
4570 RESTORE 4620
4580 FOR LOOP=1 TO 9
4590 READ ELEMENT$
4600 F$(LOOP)=ELEMENT$
4610 NEXT LOOP
4620 DATA B,F,W,7,=,J,P,5,/
4630 RANDOMIZE
4640 FOR I=1 TO 10
4650 N$=N$&" "
4660 FOR J=1 TO 5
4670 V=INT(9*RND)+1
4680 N$=N$&F$(V)
4690 NEXT J
4700 NEXT I
4710 TEXT$=N$
4720 RETURN
4730 REM  LEKTION 19:(L15+L18)

4740 CALL CLEAR
4750 PRINT "LEKTION 19"
4760 PRINT
4770 PRINT "ZEICHEN:
       ALLE!!!!"
4780 PRINT
4790 RESTORE 4840
4800 FOR LOOP=1 TO 45
4810 READ ELEMENT$
4820 F$(LOOP)=ELEMENT$
4830 NEXT LOOP
4840 DATA V,E,M,K,2,H,O,0,.,A,R,
S,6,,,I,N,8,?   ,
4850 DATA C,T,Y,3,1,L,Q,(,),Z,D,
U,4,-,G,X,9,:,B,F,W,7,=,J,P,5,/
4860 RANDOMIZE
4870 FOR I=1 TO 10
4880 N$=N$&" "
4890 FOR J=1 TO 5
4900 V=INT(45*RND)+1
4910 N$=N$&F$(V)
4920 NEXT J
4930 NEXT I
4940 TEXT$=N$
4950 RETURN
```

*Listing Morseübungsprogramm (Schluß)*

Aufbau der Lektionen:
Lektion 1: v e m k 2
Lektion 2: h o 0.
Lektion 3: vemk2 und ho0. also L1+L2

Lektion 4: a r s 6 ,
Lektion 5: i n 8 ?
Lektion 6: ars6, und in8? also L4+L5

Lektion 7: L3+L6 alle enthaltenen Zeichen

Lektion 8: c t y 3 1
Lektion 9: l q ( )
Lektion 10: cty31 und lq() also L8+L9
Lektion 11: L7+L10 alle enthaltenen Zeichen

Lektion 12: z d u 4 —
Lektion 13: g x 9 :
Lektion 14: zdu4- und gx9: also L12+L13

Lektion 15: L11+L14 alle enthaltenen Zeichen

Lektion 16: b f w 7 =
Lektion 17: j p 5 /
Lektion 18: bfw7= und jp5/

Lektion 19: L15+L18 also alle möglichen Zeichen

*Tabelle 1. Zeichenvorrat der einzelnen Lektionen*

Da die Lerngewohnheiten nicht bei jedem Menschen gleich sind, bietet das Programm zwei Methoden an. In der manuellen Ablaufart kann ein beliebiger Text mit maximal 80 Zeichen eingegeben werden. Es ist aber unbedingt darauf zu achten, daß nur zugelassene Zeichen verwendet werden. Andere Zeichen führen zum Programmabbruch. Vor der Ausgabe in Morse-Code erscheint der Text noch auf dem Bildschirm. Dieser Modus eignet sich auch in fortgeschrittenen Lernphasen als ergänzendes Training.

### *In neunzehn Lektionen den vollen Morse-Zeichensatz erlernen*

Die zweite Möglichkeit besteht darin, daß der Anwender aus 19 vorgegebenen Lektionen (Tabelle 1) eine auswählt. Unter Zuhilfenahme des Zeichenvorrats der jeweiligen Lektion stellt der Computer dann selbständig zehn verschiedene 5er-Gruppen zusammen und macht sie hörbar. Auch diese Gruppen werden zusätzlich auf dem Bildschirm ausgegeben.

Der Zeichenvorrat entspricht — ebenso wie die Einteilung der Lektionen — den Kursen des DARC und enthält 26 Buchstaben, 10 Ziffern und 9 Sonderzeichen. Ein Strich des Codes dauert in der Wiedergabe dreimal so lange wie ein Punkt. Das Morsetempo ist auf 45 Zeichen pro Minute eingestellt, so daß es auch für Anfänger langsam genug abläuft. Für Fortgeschrittene kann die Geschwindigkeit der Ausgabe erhöht werden. Dazu ist in den Zeilen 630 und 650 das Zeitargument kleiner zu wählen. Ein Verhältnis von 1 zu 3 zwischen Punkt und Strich sollte aber gewahrt bleiben. Die Zeilen 670 und 690 regeln die Pausen zwischen den Zeichen, beziehungsweise den Worten und Gruppen.

(René Harnisch)

## Nachhall

In dieser Rubrik werden zukünftig auf Anregung unserer Leser hin alle Nachbesserungen, Verbesserungen und kleine Erweiterungen zu Programmen abgedruckt, die in zurückliegenden Ausgaben von Happy Computer erschienen sind. Diese Rubrik wird immer dann zu finden sein, wenn für die betreffende Ausgabe Verbesserungsvorschläge bei uns eingegangen sind.

Leider lassen sich Fehler nie ganz vermeiden. Wir bitten Sie dafür um Entschuldigung und hoffen auf Ihr Verständnis. An dieser Stelle möchten wir uns übrigens bei all jenen Lesern bedanken, die uns auf solche Fehler hinweisen oder Verbesserungsvorschläge schikken. Sie helfen damit nicht nur uns, sondern allen unseren Lesern.

### Reaktionsvermögen steigern — ZX81 hilft dabei

(Ausgabe 12/83, Seite 65 f)

Zeile 10 des Hilfsprogramms auf Seite 65 muß lauten:

10 FOR A=16514 TO 16965

Der Hinweis zur Fehlersuche auf Seite 66 muß lauten: Im Basic-Teil ersetzen Sie dann die Zeile 30 durch »30 PRINT A;"—"; PEEK A«. Die Zeilen 20 und 40 sind zu löschen. Mit RUN...

### Solo für Sie — und Ihren TI

(Ausgabe 12/83, Seite 50 f)

Zeile 220 muß richtig lauten: 220 IF K>32 THEN 230

Zeile 2170 muß richtig lauten: 2170 IF C=68 THEN 2180 ELSE 2400

### Bildschirmroutinen für die totale Kontrolle

(Ausgabe 1/83, Seite 46)

Leider wurde ein Teil des Listings beim Druck vergessen. Hier folgen deshalb die fehlenden Zeilen:

# Idee sucht Ausführung

**Manch eine gute Programm-Idee wartet ein Leben lang darauf, ausgeführt zu werden...**
**...und mancher Programmierer wartet ein Leben lang auf eine gute Idee. Wir wollen beide zusammenbringen.**

Vielleicht haben gerade **Sie** die Programmidee, nach der alle Welt sucht, aber keine Zeit für die Ausarbeitung. Diese Idee wollen wir unkommentiert in einer Ideenliste mit Namen und Adresse veröffentlichen.

Vielleicht haben **Sie** gerade keine Idee, was Sie programmieren könnten, aber es juckt Ihnen in den Fingern und Zeit wäre auch gerade. Wenn Sie sich für eine dieser Ideen interessieren, schauen Sie in Zukunft in unsere Ideenecke. Sollten Sie dann noch weitere Informationen für die Umsetzung in ein Programm brauchen, wenden Sie sich bitte an den jeweiligen Ideenlieferanten.

Außerdem gibt es ein paar **Spielregeln**, die jeder, der an dieser Aktion teilnimmt, aus Gründen der Fairneß einhalten sollte (wir sind übrigens der Meinung, daß alle unsere Leser fair sind und überwachen die Einhaltung nicht!).

### Wer eine Idee hat

★ schickt uns diese Idee, mit oder ohne nähere Erläuterung, auf einer Karte oder in einem Brief, mit Namen und Adresse versehen

★ erhält im Falle einer erfolgreichen Ausarbeitung durch einen anderen Leser, wenn dessen Programm in Happy Computer veröffentlicht wird, für die Idee ein Honorar über 50 Mark.

**Einsendungen an**

**Happy Computer**
**Aktion Ideenecke**
**Hans-Pinsel-Str. 2**
**8013 Haar bei München**

Die Redaktion übernimmt für etwaige Folgen einer Ideenverwertung keine Haftung.

### Wer eine Idee aufgreift

★ und in ein Programm einbaut, erwähnt neben dem eigenen Namen und der eigenen Adresse den Namen und die Adresse des Ideenanbieters in der Kopfzeile des Programms oder der entsprechenden Subroutine,

★ schickt eine Kopie des Programms an den Ideenanbieter (möglichst auf Datenträger) zur freien Verwendung,

★ schickt an die Redaktion entweder ein gut lesbares Listing mit Programmbeschreibung (2zeilig, mit 50 Zeichen pro Zeile) zur Veröffentlichung (Honorar mindestens 100 Mark)

★ oder wenigstens eine Erfolgsnachricht (in diesem Fall zahlt die Redaktion dem Ideenanbieter kein Honorar und es ist Ehrensache, daß derjenige, der so ein Programm kommerziell verwertet, den Urheber der Idee am Gewinn beteiligt).

```
740 DATA 162,195,77,19,17,2,141,0,1,19,25,2,141,0,2,19,34,2,1,128,0
750 DATA 192,6,2,32,255,224,4,32,96,36,5,128,6,2,22,251,4,92,192,6,6
760 DATA 0,2,1,128,0,4,32,96,36,2,32,0,32,6,14,22,250,4,92,192,6,5
770 DATA 128,160,2,2,1,128,0,4,32,96,36,2,32,0,32,6,14,22,250,4,92
780 DATA 192,6,5,142,2,32,0,32,6,14,22,252,2,1,128,0,4,32,96,35,5
790 DATA 128,6,2,22,251,4,92,6,6,6,7,6,8,6,9,193,70,10,86,161,136
800 DATA 192,6,2,1,118,56,160,64,97,197,5,135,195,135,192,137,96,136,5
810 DATA 130,4,32,96,40,2,32,0,32,2,33,0,32,6,7,22,248,4,91,-1
820 I=31064
830 READ POKE
840 IF POKE=-1 THEN 880
850 CALL LOAD(I,POKE)
860 I=I+1
870 GOTO 830
880 CALL LOAD(28700,123,100,127,192)
890 I=32704
900 DATA 76,73,78,75,83,32,121,88,82,69,67,72,84,83,121,126,68,79,87
910 DATA 78,32,32,121,192,72,73,68,69,32,32,121,236,83,69,69,75,32,32
920 DATA 122,8,87,73,78,68,79,87,122,198,82,69,86,69,82,83,123,100
930 DATA 77,65,83,75,69,32,43,204,-1
940 READ POKE
950 IF POKE=-1 THEN 990
960 CALL LOAD(I,POKE)
970 I=I+1
980 GOTO 940
990 END
```

*Die fehlenden Zeilen des Programms »Bildschirmroutinen für die totale Kontrolle«*

## Solitaire verbessert

In Heft 12/83 wurde auf den Seiten 50 bis 52 ein Programm für den TI 99/4A vorgestellt, zu dem, wie ich meine, unbedingt einiges gesagt werden sollte. Es ist das Programm für das Spiel Solitaire, bei welchem schon beim ersten Hinsehen einige Dinge auffielen, die besser und wesentlich kürzer hätten programmiert werden können. Folgende Verbesserungsvorschläge habe ich anzubringen:

1) Grundsätzlich sollte man, wenn mit CALL HCHAR nur ein Zeichen auf den Bildschirm gebracht werden soll, die vierte Spezifikation, nämlich die Anzahl der Wiederholung gleich »1« zu setzen, weglassen. Das spart Tipparbeit und Speicherplatz.

2) Um den Spielfeldrand darzustellen, benötigt Herr Wolfgang Lenk 70 Anweisungen. Es gibt auch in TI-Basic einige wesentlich kürzere Möglichkeiten. Eine der elegantesten, die mit 22 Anweisungen auskommt, sei hier genannt. Man ersetze die Zeilen 240 bis 930 durch:

```
240 CALL CLEAR
250 CALL COLOR(12,16,16)
260 FOR I=3 TO 8
270 CALL COLOR(I,2,16)
280 NEXT I
290 DIM H(17),V(18)
300 A$="xxAxBxCxDxExFx
    Gxx"
310 B$="x1x2x3x4x5x6x7x"
320 FOR I=1 TO 17
330 H(I)=ASC(SEG$(A$,I,1))
340 NEXT I
350 FOR I=1 TO 15
360 V(I)=ASC(SEG$(B$,I,1)
370 NEXT I
380 FOR I=8 TO 24
390 CALL HCHAR(1,I,H(I-7))
400 CALL HCHAR(17,I,H(I-7))
410 NEXT I
420 FOR I=2 TO 16
430 CALL HCHAR(I,8,V(I—1))
440 CALL HCHAR(I,24,V(I—1))
450 NEXT I
```

3) Die Anweisungen in den Zeilen 960 und 1130 sind überflüssig, da die hier definierten Farben genau den Standardfarben entsprechen.

4) Für Variablen, die nur in einem Programmteil einmal verwendet werden, so zum Beispiel die Zählvariable in einer Schleife oder die Rückgabevariablen des CALL KEY-Statements, sollte man gleiche Namen vergeben. In Zeile 100 werden K und S verwendet, in Zeile 1230 dann KEY und STATUS. Um Speicherplatz zu sparen, sollte man sich seine persönlichen und möglichst kurzen Standardvariablennamen überlegen, zum Beispiel I als Schleifenzähler, K als Rückgabevariable und S als Statusvariable im CALL KEY-Statement.

5) Aufeinanderfolgende IF-Abfragen lassen sich auch in TI-Basic, wo kein logischer OR-Operator verfügbar ist, zu einer IF-Abfrage zusammenfassen. Man führt eine logische Variable ein, deren Wert nur 0 und —1 annimmt. Die Zeilen 1520 bis 1670 lassen sich so schreiben:

```
1520
L=(A=65)+(A=66)+(A=70)
+(A=71)
1530 IF L=0 THEN 1600
1560
L=(B=49)+(B=50)+(B=54)
+(B=55)
1570 IF L THEN 1210
1600
L=(C=65)+(C=66)+(C=70)
+(C=71)
1610 IF L=0 THEN 1680
1640
L=(D=49)+(D=50)+(D=54)
+(D=55)
1650 IF L THEN 1210
1680 ...
```

Siehe hierzu auch Handbuch, Seite 59.

6) In einem IF-Statement, in dem THEN zur nachfolgenden Zeile verzweigt und ELSE zu einer beliebigen Stelle im Programm, sollte der numerische Ausdruck negiert und die Verzweigung entsprechend umgestellt werden. Zum Beispiel statt

```
1700 IF D=B+2 THEN 1710
ELSE 1720
```

schreibt man besser

```
1700 IF D<>B+2 THEN 1720
```

oder statt

```
1900 IF A>C THEN 1910 ELSE
1930
```

schreibt man

```
1900 IF A<=C THEN 1930
```

Daniel Rüster

## Oric-Tips

Ich hoffe, daß folgende Tips für die Oric-1-(48 K)Benutzer interessant sein können:

Call # F42B: bewirkt einen absoluten Kaltstart (Speicher wird gelöscht, Betriebssystem wird neu initialisiert).

Call # E6CA: führt zum Abschalten der Tastatur. Durch den Wegfall der Tastaturabfrage wird der Oric zirka 25 Prozent schneller.

Call # E804: schaltet die Tastatur wieder ein. Beide Befehle (Call # E6CA und Call # E804) können innerhalb eines Programms ohne Probleme angewandt werden.

Ein Beispiel:

```
10 Call # E6CA
20 For n = 1 to 1000: Next
30 Call # E804
```

In diesem Zusammenhang ist auch die Adresse 623 (# 26F) nützlich, da diese die Gesamtanzahl der darzustellenden Zeilen beinhaltet. Diese Adresse sinnvoll zu verändern, ist sehr einfach, da man direkt die Gesamtanzahl der Zeilen als Wert »einpoken« kann. So bewirkt zum Beispiel »Poke 623,10«, daß der Oric nur noch 10 Zeilen zum Darstellen verwendet. Poke 623,27 stellt wieder den Normalzustand her. Durch sinnvolle Veränderung beider Adressen ergeben sich viele Möglichkeiten, den Bildschirm in verschiedene Fenster einzuteilen. Schließlich noch zwei neue, schon einprogrammierte Geräusche:

```
Call # FB12 = »weißes Rau-
schen«
Call # FB03 = Tastaturklick
```

Philipp Neuffer

## Stichwort: Kartei für Spectrum verbessert

Ich habe mich sehr über das Karteiprogramm für den ZX Spectrum aus Ihrer Dezember-Ausgabe gefreut. Es hat leider einen kleinen, unschönen Fehler: Beim Schreiben des Textes werden Wörter nicht getrennt. Ich habe versucht, diesen Kunstfehler zu beheben und bin zu diesem Ergebnis gekommen:

abändern:

```
540 FOR i=1 TO z: IF n$(r(i))=u$
THEN CLS: PRINT AT 0,0;
n$(r(i)),AT 1,0;x$: GOSUB 1000:
RETURN: GO TO 570
...
640 PRINT AT 0,0;n$(r(i)),AT
1,0;x$: GOSUB 1000
```

neu:

```
1000 LET test=0: LET lang=
LEN e$(r(i)): PRINT AT 1,0; " ";
LET kk=0
1010 IF lang-kk<31 THEN
PRINT e$(r(i))(kk+1 TO lang):
RETURN
1020 IF kk+31>lang THEN
FOR g=lang TO kk STEP —1:
GO TO 1040
1030 FOR g=kk+31 TO kk+16
STEP —1
1040 IF e$(r(i))(g)=" " OR
e$(r(i))(g)="■" THEN LET
test=1: PRINT e$(r(i))(kk+1 TO
g—1): LET kk=g: GO TO 1010
1050 NEXT g
1060 IF test<>1 THEN FOR
m=kk+15 TO kk+1 STEP —1:
IF e$(r(i))(kk+1 TO m—1): LET
kk=m: LET test=0: GO TO 1010
1070 IF test<>1 THEN NEXT m
1080 IF test<>1 THEN PRINT
e$(r(i))(kk+1 TO kk+25): LET
test=0: GO TO 1010
```

Stefan Förster

## So läuft der Spectrum ungestört

In der Dezemberausgabe von HC fragte W. Sirges nach den Ursachen einer Störung des Rundfunkempfangs durch den ZX Spectrum. Dies kann zwei Ursachen haben:

1) die Spannungswandlerspule
2) Hochfrequenzmodulator und Bildgenerator.

Die Spannungswandlerspule ist wohl eher ein »Krachmacher« als ein Störenfried, man kann sie aber auch abschirmen. In kleinen Transistorradios sieht man des öfteren kleine silberne »Kästchen«, die oben eine Öffnung haben, durch die eine Schraube zu sehen ist. Ich habe einen solchen Filterbaustein seiner Kappe beraubt und diese der Spule im Spectrum übergestülpt. Das Problem ist nur, eine ausreichend große Kappe zu finden. Ist die Kappe drüber, wird's merklich stiller, stören tut es aber immer noch. Um dem abzuhelfen, muß man das Gehäuse des ZX Spectrum innen mit Alufolie auskleiden. Doch Vorsicht! Zwischen Alufolie und Lötseite der Platine eine Isolierschicht (Plastik-70 oder ganz dünnes Papier) einfügen, da sonst die Gefahr eines Kurzschlusses besteht. Die Alufolie sollte man mit der Gerätemasse verbinden. Ich habe dieses Verfahren mit Erfolg beim ZX Spectrum und bei meinem jetzigen Computer, dem Oric-1 (der keine Spule besitzt, aber auch »stört«), praktiziert. Noch ein Tip: Um dem Spectrum, der ja ein größeres Kühlblech als der ZX 81 hat, das Heizen abzugewöhnen und die Spule zum Schweigen zu bringen, sollte man den Spannungsregler auslöten und überbrücken. Dann muß man ein Netzteil verwenden, welches geregelte 5 V Gleichstrom bei 1,5 A abgibt. Dieses Netzteil kann man leicht selbst bauen, es ist nur auf eine genügend große Ausgangskapazität zu achten. Ergebnis der ganzen Behandlung: Ein kalter, stiller Computer, der nur dann »Störendes« von sich gibt, wenn man ihm das einprogrammiert. Wer sich nicht zutraut, ein solches Netzgerät zusammenzustellen, kann von mir einen Schaltplan (gegen Rückporto) bekommen. Dietmar Belloff

## Hier sind Clubs

Erfahrungsaustausch sowie kostenloser Software-Tausch sind Ziele des neu gegründeten TI 99/4A-Tauschclub Ruhrgebiet. Der Club verfügt über Programme im Gesamtvolumen von rund 1 MByte. Wir beantworten gern Anfragen, wenn Rückporto beigefügt ist. Ein Clubbeitrag wird nicht erhoben. Kontaktadresse: Weberstr. 10, 4300 Essen 1.

Rainer Frehse

Für Benutzer der Computer Commodore 64 und Apple II haben wir einen Programm-

# Abenteuer vor Madagaskar

**Neben der Badewanne kann Ihnen mit dem hier vorgestellten Programm in Zukunft auch Ihr Fernsehgerät als »Übungsgewässer« dienen. »Vor Madagaskar« ist für die TI99/4A-Konsole und Extended-Basic geschrieben. Es stellt eine moderne Version des Spiels »Schiffchen versenken« dar.**

Die Spielhandlung findet in einem schematisierten Meer statt, welches in neun quadratische Seegebiete unterteilt ist. In dreien davon befinden sich feindliche Konvois, die aufgespürt und angegriffen werden sollen. Der Spieler muß das angreifende U-Boot steuern.

Zu Beginn des Spiels ist das U-Boot getaucht (Blick auf vorbeischwimmende Fische). Es wird der eigene Standort angezeigt und der Standort einer der drei gegnerischen Flotten.

TI 99/4A

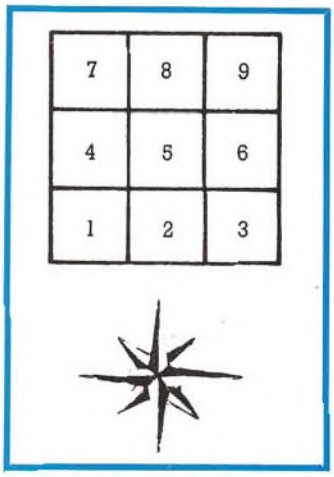
```
25 DIM TYP(1Ø),V(1Ø),LS(16)
35 CALL CLEAR :: RANDOMIZE ::
   OO=INT(RND*9)+1 ::TA=4 :: MUN=99

Zeichen, Farben:

37 FOR K=1 TO 17 :: READ I,Z$ :: CALL CHAR(I,Z$):: NEXT K

39 DATA 34,ØØØØ Ø87D FF7D ØØØØ  ØØØØ 1ØBE FFBE,
        4Ø,ØØØØ 281Ø 7C1Ø 28ØØ  Ø,42,Ø,43,Ø,
        6Ø,Ø1Ø4 2115 Ø825 Ø5Ø2  Ø7ØA 11Ø4 ØBØ3 Ø7Ø7
           4ØA8 8Ø24 8192 6ØC8  54DØ 2ØDØ EØA8 EØE
41 DATA 64,FEFE FCFC F8FØ CØØØ  ØØØ7 ØF1F 3F3F 7F7F,
        67,ØØCØ FØF8 FCFC FEFE  7F7F 3F3F 1FØF Ø3ØØ
           ØØØØ ØØØØ 7E24 FF7E  8,
        8Ø,ØØ38 4444 7C44 4444  ØØ7C 4Ø4Ø 784Ø 4Ø7C
43 DATA 86,1Ø1Ø 1ØFE 1Ø1Ø 1, 89,ØØ7E 7E7E 7E7E 7E,
        91,ØØØØ ØØØØ ØØØ8 Ø8Ø8  ØØØØ ØØØØ ØØØØ Ø1Ø1,
       124,ØØØØ 4438 447C 4444  ØØØØ 4438 4444 4438
45 DATA 132,ØØØØ ØØØØ Ø4Ø4 7E3E  ØØØØ ØØØØ 2Ø2Ø 7E7C
           ØØØØ Ø9Ø6 26FF 7F3E  ØØØØ 9Ø6Ø 64FF FE7C,
        136,ØØØØ ØØØ1 ØØFF 7F3F  Ø,
        138,ØØØØ Ø8FC 78FF FFFE  Ø
47 DATA 14Ø,ØØØØ 1Ø3F 1EFF 7F7F  Ø,
        142,ØØØØ ØØ8Ø ØØFF FEFC  Ø

6Ø K$=RPT$("F",16)::
   CALL CHAR(36,RPT$("F",64),44,K$,58,K$&"Ø",66,K$,88,K$,
             93,RPT$("Ø",64),128,RPT$("FFFFFFFF"&RPT$("Ø",24),2))

80 CALL COLOR(1,1,1, 2,1,1, 3,13,4, 4,13,4, 5,15,1, 6,2,4, 7,2,4)
81 CALL COLOR(8,10,1, 9,2,13, 1Ø,2,13, 11,2,13, 12,2,13,
              13,11,1, 14,11,1)

Maske:

9Ø CALL SCREEN(2):: CALL HCHAR(2Ø,23,65)::
   CALL HCHAR(2Ø,24,66,2):: CALL HCHAR(2Ø,26,67)
1ØØ FOR I=21 TO 22 :: CALL HCHAR(I,23,66,4):: NEXT I
1Ø5 CALL HCHAR(23,23,68):: CALL HCHAR(23,24,66,2)::
   CALL HCHAR(23,26,64):: CALL CLTAFEL ::
   CALL VCHAR(2Ø,3,89,2)
12Ø FOR I=Ø TO 3 :: READ S$(I)::
   DISPLAY AT(2Ø+I,26)SIZE(3):S$(I):: NEXT I

13Ø DATA mun,12T,99K,;;;

Spielverlauf, Steuerprogramm:

17Ø FOR I=1 TO 3
175 ORT(I)=INT(RND*9)+1 :: IF ORT(I)=ORT(I-1) THEN 175
18Ø NEXT I :: IF ORT(1)=ORT(3)THEN 17Ø ELSE CALL MUSIK
195 CALL TAFEL :: CALL VCHAR(21,21,48+OO)::
   CALL VCHAR(23,21,48+ORT(1))
2ØØ CALL DELSPRITE(ALL):: CALL TAUCH
23Ø CALL KURS(OO,K$):: IF ASC(K$)=48 THEN IF TRR=6
   THEN GOSUB 3ØØØ :: GOTO 4ØØ ELSE 4ØØ
24Ø GOSUB 3ØØØ :: TRR=TRR-1 :: RESTORE 4Ø4Ø :: CALL TAFEL
   :: CALL HCHAR(21,18,OO+48)
25Ø FOR I=659 TO 523 STEP -136 :: FOR K=5 TO 25 STEP 4 ::
   CALL SOUND(-35,I,K):: NEXT K :: NEXT I :: CALL HALT(K)
4ØØ RESTORE 4Ø3Ø :: CALL TAFEL
41Ø CALL HALT(K):: ON K GOTO 5ØØ,23Ø,41Ø,41Ø

Periskop ausgefahren:

5ØØ A=1 :: CALL DELSPRITE(ALL):: CALL PERISKOP :: VI=17
```

*(Die Leerzeichen in den Zeilen 39 bis 47 sind außer nach »DATA« nicht einzugeben und dienen nur der Übersichtlichkeit!)*

```
510 FOR I=1 TO 3 :: IF ORT(I)=00 THEN L2=INT(RND*6)+7
    :: GOTO 525
515 NEXT I
520 RESTORE 4050 :: CALL TAFEL :: CALL HALT(K)::
    U=5.1 :: ON K GOTO 800,2000,5200,5200
525 VI=17 :: TORP,L=0 :: F=4 :: NR=5 ::
    IF A=1 THEN CALL HCHAR(1,17,88)
600 IF RND < .6 THEN  GR=1 ELSE GR=INT(RND*3)+2
610 IF GR=2 THEN ZAHL=INT(RND*4)+1 ELSE ZAHL=INT(RND*2)+1
620 ZL=ZAHL :: CALL MAGNIFY(GR) ::
    IF GR=1 THEN Z$="56" ELSE IF GR=2 THEN Z$="5678"
    ELSE Z$="89"
640 CALL KEY(0,K,S):: IF S=0 THEN 680
650 ON A GOTO 660,830

660 CALL VISIER1(VI):: IF K= 32 AND T > 0 THEN TORP=1 ::
    CALL TORPEDO(GR,VI,T,TORP)
670 IF K=49 THEN 800 ELSE IF K=50 THEN 2000
680 IF ZL > 0 THEN ZL=ZL-1 :: L=1 ::
    CALL SCHIFF(GR,NR,Z$,VI,A,TYP(),V())
    ELSE L=0 :: CALL KONTAKT(TORP,ZAHL,I,GR,TYP(),TA,T,L,A,VI)

690 IF L=1 THEN 640 ELSE IF A=1 THEN CALL HCHAR(1,VI,89)
    ELSE CALL VCHAR(1,17,44,6)
700 L2=L2-ZAHL :: IF L2 > 0 THEN 525
730 FOR I=1 TO 3 :: IF ORT(I)=00 THEN  ORT(I)=0 ::
    ON A GOTO 5200,8200
740 NEXT I :: GOTO 8200

Aufgetaucht:

800 CALL DELSPRITE(#10):: CALL COLOR(1,1,1,2,1,1):: A=2 ::
    FOR I=1 TO 6 :: CALL HCHAR(I,1,44,32):: NEXT I
805 FOR I=7 TO 17 :: CALL HCHAR(I,1,36,32):: NEXT I ::
    CALL HCHAR(15,1,93,32):: FOR I=3 TO 30 STEP 9 ::
    CALL VCHAR(14,I,93,4):: NEXT I
807 CALL SPRITE(#15,128,2,113,104,#16,128,2,113,136)
810 CALL COLOR(1,5,2,2,6,4):: RESTORE 4080 :: CALL TAFEL  ::
    CALL COLOR(5,10,1):: CALL VCHAR(4,17,86)
815 IF U=5.1 THEN U=0 :: GOTO 825 ELSE 680
820 CALL VCHAR(1,17,44,5):: RESTORE 4080 :: CALL TAFEL
825 CALL HALT(K):: ON K GOTO 825,850,2000,9000
830 CALL VISIER2(F,ZAHL,V())
840 IF K=32 AND MUN > 0 THEN CALL KANONE(F,MUN,ZAHL,TA,GR,
    I,U3,TYP(),V())
850 CALL KEY(0,K,S):: IF K=50 THEN A=1 :: CALL PERISKOP
    :: GOTO 680 ELSE IF K=52 THEN 9000 ELSE IF K=51 THEN 2000
    ELSE 680

Funkpeilung:

900 CALL MAGNIFY(1):: CALL VCHAR(1,17,44,6):: I=0
905 CALL SPRITE(#2,110,10,142,189,#3,111,10,163,209,
    #4,115,10,185,189,#5,119,10,163,169,#1,70,2,168,206)
910 IF I > 15 THEN I=0 ELSE IF I < 0 THEN I=15
920 CALL KEY(0,K,S):: IF S<>0 THEN 940
930 CALL SOUND(40+INT(RND*2)*80,600,LS(I),-6,MIN(LS(I)+4,30))
    :: GOTO 910
940 IF K=122 THEN I=I+1 :: GOTO 960 ELSE IF K=46 THEN I=I-1
    :: GOTO 960
950 IF K=51 THEN 2000 ELSE IF K=49 THEN 970 ELSE 930
```

*Listing zu »Madagaskar«. Die Zwischenüberschriften sind nicht mit einzugeben*

Nach entsprechender Aufforderung im Anzeigenfeld unter dem Sichtfeld wird der gewählte Kurs eingegeben und zwar als Himmelsrichtung (n, s, o, w, no, so, nw, sw und 0, sämtliche Eingaben in Kleinbuchstaben). Das U-Boot bewegt sich immer um ein Seegebiet weiter, wie auf einem Schachbrett, also auch diagonal. Betrachten wir ein Beispiel. Um vom eigenen Standort 1 nach Gebiet 8 zu gelangen, müssen zwei Kurseingaben erfolgen, »no« und »n«, wobei die Reihenfolge beliebig ist. Die Eingabe 0 bedeutet »Stehenbleiben« beziehungsweise eine nur kurze Fahrt innerhalb des Aufenthaltsgebietes zum Aufspüren der in diesem Gebiet operierenden Schiffe. Positionsveränderungen sind nur im getauchten Zustand möglich. Nach jeder Fahrt erfolgt eine Anzeige, die darüber informiert, welches Gebiet erreicht wurde. Von dieser Anzeige muß mit Taste »1« auf die Eingabe von Aktionen umgeschaltet werden. Über die möglichen Handlungen des Spielers informiert das Anzeigenfeld. Aus dem getauchten Zustand kann man zunächst auf Periskop-Höhe gehen und von dort aus ganz auftauchen.




***Aufteilung des »Seegebiets« am Fernsehschirm***

Wenn sich im Aufenthaltsgebiet Schiffe befinden, so werden diese von den Instrumenten des U-Bootes schon wahrgenommen, bevor sie im Sichtfeld des Seerohres erscheinen. Es ertönt dann ein Warnton. Eine Information über die Richtung,

aus der die Schiffe kommen, wird zusätzlich angezeigt. Über dem Sichtfeld befindet sich ein Visier, welches mit den Tasten »z« und ».« verstellt werden kann. Mit der Leertaste wird ein Torpedo freigegeben, dessen Marschrichtung von der Visiereinstellung bestimmt ist. Torpedotreffer versenken ein Schiff in jedem Fall.

Wenn keine weitere Schiffswarnungen gegeben werden und die Aktionsanzeige erscheint, dann befinden sich keine gegnerischen Schiffe mehr im Aufenthaltsgebiet. Sämtliche Visiereinrichtungen werden dann automatisch abgeschaltet und sind auf die momentanen Werte fixiert.

### Tauchübungen mit Zielwasser

Im aufgetauchten Zustand können keine Torpedos mehr abgefeuert werden, sondern nur die Bordkanone, deren Visierkreuz mit den Tasten »=« und »/« aufgerufen und verstellt wird. Die Höhe dieses Fadenkreuzes bestimmt den vertikalen Winkel des Feuers und damit die Entfernung, in der die unsichtbaren Granaten niedergehen. Diese muß auf die Entfernung des ins Visier genommenen Schiffes abgestimmt werden. Den richtigen Zusammenhang bekommt man durch Probieren heraus. Ein Treffer wird dann erzielt, wenn — bei richtiger Höheneinstellung — ein Schiff sich im Moment des Auftreffens der Granate direkt unter dem Visierkreuz befindet.

Horizontale Schwenks erfolgen mit den Visiertasten »z« und ».«. Dabei dreht sich die gesamte Visieroptik im Kreis, solange eine der Tasten gedrückt ist. Es entsteht der Eindruck, als ob die gesichteten Schiffe vorwärts und rückwärts verschoben werden. Ihre Eigengeschwindigkeit und Bewegung relativ zueinander bleibt dabei erhalten; es kommt nur die Schwenkgeschwindikeit der Zieloptik hinzu. Die Kanone feuert, solange die Leertaste gedrückt wird.

Treffer haben unterschiedliche Wirkungen, die von der Genauigkeit der

```
96Ø CALL LOCATE(#1,168-INT(14*SIN(PI/8*I)+.5),192+INT(14*
    COS(PI/8*I)+.5)):: GOTO 91Ø
97Ø FOR I=1 TO 5 :: CALL DELSPRITE(#I) :: NEXT I :: GOTO 83Ø

Speicherbelegung für Funkpeilung:

3ØØØ ORT(Ø)=00 :: FOR I=Ø TO 3 ::
     IF ORT(I)< 4 THEN X(I)=ORT(I) :: Y(I)=1 :: GOTO 3Ø2Ø
3Ø1Ø IF ORT(I)< 7 THEN X(I)=ORT(I)-3 :: Y(I)=2
     ELSE X(I)=ORT(I)-6 :: Y(I)=3
3Ø2Ø NEXT I
3Ø3Ø FOR I=1 TO 3 ::
     X(I)=X(I)-X(Ø) :: Y(I)=Y(I)-Y(Ø) :: NEXT I ::
     X(Ø),Y(Ø)=Ø
3Ø4Ø FOR I=Ø TO 15 :: LS(I)=3Ø :: NEXT I


3Ø5Ø FOR I=1 TO 3 :: IF ORT(I)=Ø OR ORT(I)=00 THEN 31ØØ
3Ø6Ø IF X(I)=Ø THEN IF Y(I)> Ø THEN W=4 ELSE W=12
     ELSE 3Ø8Ø
3Ø7Ø GOTO 3Ø9Ø
3Ø8Ø W1=INT(ATN(Y(I)/X(I))*8/PI+.5)::
     IF X(I)< Ø THEN W=8+W1 ELSE
     IF Y(I)< Ø THEN W=16+W1 ELSE W=W1
3Ø9Ø LS(W)=4

31ØØ W1=W+1 :: IF W1 > 15 THEN W1=Ø
311Ø W2=W1+1 :: IF W2 >15 THEN W2=Ø
312Ø W3=W-1 :: IF W3< Ø THEN W3=15
313Ø W4=W3-1 :: IF W4 < Ø THEN W4=15
314Ø LS(W1)=MIN(LS(W1),15):: LS(W3)=MIN(LS(W3),15)::
     LS(W2)=MIN(LS(W2),26):: LS(W4)=MIN(LS(W4),26)
315Ø NEXT I :: RETURN

Texttafeln:

4Ø2Ø DATA du `befindest `dich,im ` seegebiet,
        aufkl¦rung ` meldet,feindverband ` in
4Ø3Ø DATA PKTIONQN;;;;;;;;;,1 ` periskop,2 ` weiterfahren,`
4Ø4Ø DATA  `, ``` seegebiet, ``` erreicht,`
4Ø5Ø DATA PKTIONQN;;;;;;;;;,1 ` auftauchen,2 ` tauchen,`
4Ø6Ø DATA ` , ` ,dort `ist `das `ufer,dussel
4Ø65 DATA feindschiff,steuerbord ` voraus,`,`
4Ø7Ø DATA feindschiff,backbord `voraus,` ,`
4Ø75 DATA  `,`` gegner ` wurde,````  zerst}rt,`
4Ø8Ø DATA 1 ` kanone ` klar,2 ` periskop,3 ` tauchen,
        4 ` funkpeilung
4Ø9Ø DATA ;;PHOI;; ` du `wirst,bef}rdert `und,
        bekommst ` einen,orden


Unterprogramme:

5ØØØ SUB ZEIT(T)
5Ø1Ø FOR DELAY=1 TO T :: NEXT DELAY :: SUBEND


5Ø3Ø SUB PERISKOP
5Ø4Ø CALL COLOR(1,1,1,2,1,1):: CALL CLTAFEL ::
     CALL HCHAR(1,4,89,28):: CALL DELSPRITE(#15,#16)
5Ø42 FOR I=1 TO 3 :: CALL VCHAR(1,I,32,17):: NEXT I
     CALL VCHAR(1,32,32,17):: FOR I=14 TO 17 ::
     CALL HCHAR(I,4,36,28):: NEXT I
```

```
5045 FOR I=2 TO 6 :: CALL HCHAR(I,4,44,28):: NEXT I
     CALL COLOR(1,5,2,2,6,4) :: RESTORE 4050 ::
     CALL TAFEL :: SUBEND


5070 SUB ERFOLG
5080 RESTORE 4075 :: CALL TAFEL ::
     FOR I=1 TO 3 :: CALL SOUND(80,188+77*I-3*I*I,2):: NEXT I
     :: CALL SOUND(300,523,2,262,5,659,5):: SUBEND

5100 SUB SCHIFF(GR,NR,Z$,VI,A,TYP(),V())
5120 IF Z$="" THEN SUBEXIT
5130 IF LEN(Z$)=1 THEN ZEI=VAL(Z$):: Z$="" :: GOTO 5160
5140 P=INT(RND*LEN(Z$))+1 :: ZEI=VAL(SEG$(Z$,P,1))::
     Z$=SEG$(Z$,1,P-1)&SEG$(Z$,P+1,LEN(Z$)-P)
5160 ZEILE=ZEI*8+1 :: NR=NR+1 :: V(NR)=ZEI-INT(RND*3)::
     IF RND<.5 THEN V(NR)=-V(NR)
5165 IF GR>2 THEN TYP(NR)=140-(1-SGN(V(NR)))*2 :: GOTO 5220
5167 IF RND<.1 THEN TYP(NR)=69 :: GOTO 5220
5170 ON ZEI-4 GOTO 5180,5180,5190,5200
5180 IF RND<.5 THEN TYP(NR)=133-(1-SGN(V(NR)))/2 :: GOTO 5220
5190 TYP(NR)=135-(1-SGN(V(NR)))/2 :: GOTO 5220
5200 IF RND<.5 THEN 5190 ELSE TYP(NR)=69
5220 IF NR>7 THEN 5290
5240 IF NR>6 THEN DISPLAY AT(22,3)SIZE(17):"weitere einheiten"
     :: DISPLAY AT(23,3)SIZE(11):"im anmarsch" :: GOTO 5290
5252 FOR I=1 TO 2 :: CALL SOUND(250,-3,2):: CALL ZEIT(160)
     :: NEXT I
5253 IF SGN(V(NR))=1 THEN RESTORE 4070 ELSE RESTORE 4065
5255 CALL TAFEL
5270 FOR I=1 TO 30 :: IF A=1 THEN CALL VISIER1(VI)
     ELSE CALL VISIER2(F,ZAHL,V())
5280 NEXT I
5290 CALL SPRITE(#NR,TYP(NR),2,ZEILE,256,0,V(NR))
     :: CALL ZEIT(200):: SUBEND
5300 SUB KONTAKT(TORP,ZAHL,I,GR,TYP(),TA,T,L,A,VI)
5320 FOR I=6 TO 5+ZAHL :: CALL POSITION(#I,Y,X)::
     IF Y=0 THEN 5350
5330 L=1 :: IF X<5 OR X>250 THEN CALL DELSPRITE(#I)
     :: CALL CLTAFEL
5335 IF A=2 THEN 5350 ELSE IF GR=1 OR GR=3 THEN
     TOL=5 ELSE TOL=6+GR/2
5340 IF TORP=0 THEN SUBEXIT
     ELSE CALL COINC(#I,#10,TOL,CO):: IF CO=-1 THEN TORP=0
     :: CALL TREFFER(I,GR,TYP(),V(),T,TA,U3,VI,A)
5350 NEXT I :: IF A=2 THEN SUBEXIT
5370 CALL POSITION(#10,Y,X):: IF Y<35 THEN
     CALL DELSPRITE(#10):: TORP=0 :: SUBEXIT
5375 L=1 :: SUBEND


5400 SUB TORPEDO(GR,VI,T,TORP)
5410 FOR I=0 TO 27 STEP 3 :: CALL SOUND(-30,220+20*I,15,-7,I)
     :: NEXT I
5420 TORP=1 :: IF GR=1 OR GR=3 THEN Y=129 ELSE Y=117
5430 IF GR=1 THEN X=VI*8-7 ELSE IF GR=3 THEN X=VI*8-11
     ELSE X=VI*8-17
5440 IF GR=1 THEN TYP=91 ELSE TYP=92
5460 CALL SPRITE(#10,TYP,6,Y,X,-4,0):: T=T-1 ::
     I=LEN(STR$(T)):: IF I=1 THEN CALL HCHAR(21,28,59)
5465 DISPLAY AT(21,28-I)SIZE(I):STR$(T):: SUBEND
5500 SUB TREFFER(I,GR,TYP(),V(),T,TA,U3,VI,A)
```

*Listing zu »Madagaskar« (Fortsetzung) Die Zwischenüberschriften sind nicht mit einzugeben*

**Zeile 90-130:** Aufbau der Bildschirmmaske (»Instrumente«, Anzeigentafel)
**170-195:** Auswürfeln der Schiffsaufstellung des Gegners (drei verschiedene Ziffern); ein Standort wird bekanntgegeben.
**200-410:** Aufruf des Tauch-Programms (Blick unter Wasser), Wahl des Kurses. Die Pause, die jede Fahrt simuliert, wird genutzt, um im Unterprogramm mit den Zeilen 3000 bis 3150 die relative Lage der feindlichen Konvois gegenüber dem eigenen Standort zu berechnen.
**500-525:** Aufruf des Periskop-Programms (Blick durch Sehrohr, Angriffe mit Torpedo). Zufallsentscheidung, wieviele (L2) Schiffe auftauchen werden, Anfangswerte
**600-740:** Zentraler Teil des Programms, steuert die Reihenfolge, in der die Unterprogramme aufgerufen werden. Es wird die vorgesehene Anzahl Schiffe erzeugt, bis die Gesamtzahl L2 einer Runde erreicht ist. Solange keine Tastatureingaben zur Betätigung der Visiere oder der Wahl anderer Aktionen erfolgen, kontrolliert das Programm die Sprites. Der String Z$ enthält die möglichen Nummern der Zeilen, auf denen sich die Schiffssprites bewegen dürfen.
Drei Indikatoren bewirken Verzweigungen, die vom vorausgegangenen Spielablauf abhängen:
A entscheidet, ob Spielhandlungen und Kontrollen für Unterwasseraktionen (A=1) erfolgen oder die Aktionen des aufgetauchten Zustandes (A=2).
L zeigt an, ob gerade Schiffssprites existieren (L=1) oder — falls nicht (L=0) — die Kontrollroutine im Kontakt-Unterprogramm abgekürzt werden kann.
U zeigt an, ob sich im Aufenthaltsgebiet noch gegnerische Schiffe befinden (U=0) oder — falls nicht (U=5,1) — die Visiereinrichtungen der Kanone abgeschaltet werden.
**800-850:** Aktionen des aufgetauchten Zustandes. Der Sprung nach Zeile 820 erfolgt, wenn sich im Aufenthaltsgebiet keine Schiffe mehr befinden.
(807) Die Sprites 15 und 16 überdecken die Mitte der Reling, die sonst vom »unsichtbaren« Sprite 28 durchbrochen werden könnte.
**900-970:** Unterprogramm Funkpeilung; der Peilsprite 1 wird mit der Sinus- und Cosinusfunktion in Zeile 960 innerhalb des kleinen Bildschirms im Kreis bewegt.
(930) erzeugt Morsegeräusche in den Lautstärken LS(I), die vom Unterprogramm 3000-3150 berechnet wurden.

*Erläuterungen zum Programm »Vor Madagaskar«*

**Felder:**

**TYP( ):** ASCII-Codes der Schiff-Sprites
**V( ):** Geschwindigkeit der Schiffe
**LS( ):** Lautstärken der Funkpeil-Richtungen
**ORT( ):** Seegebiete (Ziffern 1 bis 9) mit feindlichen Konvois
**X( ),Y( ):** Seegebiete in kartesischen Koordinaten
**S$( ):** Datensätze für Tafelausdrucke

**Wichtige Variablen des Hauptprogramms:**
**Z$:** String mit den möglichen Zeilennummern der zufällig zu erzeugenden Schiffe
**K$:** eingeschlagener Kurs, z.B. »no«, »w«, »0«
**T,MUN:** Torpedo- bzw. Kanonenmunition
**TORP:** Indikator dafür, ob Torpedos unterwegs
**TA:** Trefferanzeige, Spalte des nächsten anzuzeigenden Schiffsymbols
**GR:** Magnify-Faktor
**VI,F:** Spalte der Visiermarkierungen für Torpedo und Kanone
**W,W1:** Winkel des Funkpeilstrahls (ganzzahlige Bogenmaßwerte, bezogen auf Vollwinkel 16)
**W1 ... W4:** benachbarte Winkel
**OO:** eigener Standort
**L,L2:** Zähler für Entscheidung, ob weitere Schiffe aufgerufen werden sollen; L2 = Zahl der Schiffe im Aufenthaltsgebiet
**Zahl:** Anzahl der gleichzeitig auftauchenden Schiffe
**ZL:** Anzahl der noch zu erzeugenden Schiffe
**A:** Indikator für Aktionsart Torpedo/Kanone
**TRR:** Indikator für Notwendigkeit einer Ortsberechnung für Funkpeilung

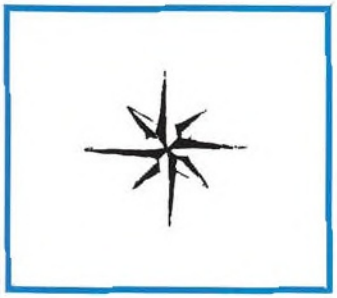
**Wichtige lokale Variablen in den Unterprogrammen:**
**NR:** (Sprite-)Nummern gleichzeitig erzeugter Schiffe (6...9)
**ZEI:** zufällig gewählte Zeile für erzeugtes Schiff, abhängig vom Schiffstyp und GR, ZEILE = entsprechende Dot Row
**X1,Y1:** Koordinaten des unter ein sinkendes Schiff gelegten »unsichtbaren« Sprites
**X2,Y2:** Koordinaten des Explosionssprites
**E:** Indikator für erstmaligen Durchlauf des TAUCH-Programms
**U2,F:** Zielparameter für Kanonenvisier
**U3:** Indikator für Treffer und Zerstörungsrate bei Kanone, abhängig von Abstand Schiffsmitte-Visierspalte und Schußzahl

***Variablenliste zu »Madagaskar«***

```
5510 CALL MOTION(#I,0,0):: CALL DELSPRITE(#10)
     :: TORP=0 :: Q=TYP(I)
5515 CALL POSITION(#I,Y,X):: CALL SPRITE(#28,Q,2,Y,X)::
     CALL DELSPRITE(#I):: IF A=1 THEN X1=VI*8-7 :: Y1=Y+3
     ELSE X1=X :: Y1=Y
5520 CALL SPRITE(#5,40,10,Y1,X1):: CALL ZEIT(2):: CALL DELSPRITE(#5)
5525 FOR J=0 TO 28 STEP 4 :: CALL SOUND(-50,110,J,117,J,
     139,J,-6,J):: NEXT J
5527 IF U3>0 THEN CALL SPRITE(#I,Q,2,Y,X)::
     CALL DELSPRITE(#28):: V(I)=INT(V(I)/2)::
     CALL MOTION(#I,0,V(I)):: SUBEXIT
5530 X1=MAX(X,25):: ON GR GOTO 5540,5550,5560,5570
5540 X1=MIN(X1,241):: Y1=Y+8 :: X2=X-3 :: Y2=Y-2 :: GOTO 5590
5550 X1=MIN(X1,233):: Y1=Y+16 :: X2=X-6 :: Y2=Y-4 :: GOTO 5590
5560 X1=MIN(X1,233):: Y1=Y+8 :: X2=X :: Y2=Y-11 :: GOTO 5590
5570 X1=MIN(X1,217):: Y1=Y+16 :: X2=X :: Y2=Y-22
5590 CALL SPRITE(#4,60,2,Y2,X2):: FOR J=0 TO 27 STEP 3 ::
     CALL SOUND(-60,110,J,-7,J):: NEXT J ::
     CALL DELSPRITE(#4)
5610 CALL SPRITE(#27,36,5,Y1,X1)::
     FOR K=Y TO Y+(Y1-Y)*.75 :: CALL ZEIT(15)::
     CALL LOCATE(#28,K,X):: NEXT K
5620 CALL DELSPRITE(#28,#27):: CALL ERFOLG
5640 IF Q>135 THEN 5650 ELSE IF Q=133 OR Q=135 THEN Q=Q-1
5645 CALL HCHAR(18,TA,Q):: TA=TA+2 :: GOTO 5657
5650 IF Q=140 THEN Q=136
5655 FOR I=0 TO 1 :: CALL HCHAR(18,TA+I,Q+2*I):: NEXT I
     :: TA=TA+3
5657 IF TA>30 THEN CALL MUSIK :: RESTORE 4090
     :: CALL TAFEL :: CALL HCHAR(18,1,32,32)::
     CALL VCHAR(20+0,32,34):: O=O+1 :: TA=4 :: SUBEND
5700 SUB VISIER1(VI)
5705 CALL KEY(0,K,S):: IF S=0 THEN SUBEXIT
5710 IF K=122 AND VI>4 THEN CALL HCHAR(1,VI,89)::
     VI=VI-1 ELSE IF K=46 AND VI<31 THEN
     CALL HCHAR(1,VI,89) :: VI=VI+1 ELSE SUBEXIT
5720 CALL HCHAR(1,VI,88):: GOTO 5705
     :: SUBEND
5740 SUB VISIER2(F,Zahl,V())
5750 CALL KEY(0,K,S):: IF S=0 THEN SUBEXIT
5760 IF K=122 THEN V1=20 ELSE IF K=46 THEN V1=-20
     ELSE 5800
5770 FOR I=6 TO 5+ZAHL :: CALL MOTION(#I,0,V(I)+V1)
     :: NEXT I
5780 CALL KEY(0,K,S):: ON S+2 GOTO 5780,5790,5750
5790 FOR I=6 TO 5+ZAHL :: CALL MOTION(#I,0,V(I))
     :: NEXT I :: GOTO 5750
5800 IF K=47 AND F<5 THEN F=F+1 ELSE
     IF K=61 AND F>1 THEN F=F-1 ELSE SUBEXIT
5810 CALL VCHAR(1,17,44,5):: CALL VCHAR(F,17,86)::
     GOTO 5750 :: SUBEND
5830 SUB TAUCH
5835 CALL HCHAR(1,3,32,29):: CALL MAGNIFY(1)::
     CALL COLOR(1,1,1,2,1,1)
5840 FOR I=3 TO 17 :: CALL HCHAR(I,1,36,32):: NEXT I
     :: CALL COLOR(1,6,2,5,15,1)
5845 IF E=0 THEN K=28 ELSE K=INT(RND*5)+4
5850 FOR I=2 TO K :: IF RND<.5 THEN F=-1 ELSE F=1
```

```
5855 V=(INT(RND*4Ø)+1Ø)*F ::
     CALL SPRITE(#I,35-(1-F)/2,INT(RND*13)+3,INT(RND*112)+17,
     256,Ø,V)
586Ø NEXT I :: E=1 :: SUBEND
588Ø SUB HALT(K):: DISPLAY AT(24,1)BEEP
5885 CALL KEY(Ø,K,S):: IF S=Ø OR K<49 OR K>52 THEN 5885
     ELSE K=K-48
589Ø SUBEND
592Ø SUB CLTAFEL ::
     FOR I=2Ø TO 23 :: CALL HCHAR(I,5,96,17):: NEXT I
     :: SUBEND
595Ø SUB KANONE(F,MUN,ZAHL,TA,GR,I,U3,TYP(),V())
596Ø U1=Ø
597Ø IF U1=MUN THEN 599Ø ELSE CALL SOUND(5Ø,11Ø,Ø,147,1,196,3,-7,1)
     :: U1=U1+1
598Ø CALL KEY(Ø,K,S):: IF K=32 THEN 597Ø
599Ø MUN=MUN-U1 :: I= LEN(STR$(MUN))::
     IF I=1 THEN CALL HCHAR(22,28,59)
5995 DISPLAY AT(22,28-I)SIZE(I):STR$(MUN)
6Ø2Ø FOR I=6 TO 5+ZAHL :: CALL POSITION(#I,Z,S)::
     U3=INT((ABS(S-129+INT(GR/2)*8))/(RND*U1+1))
6Ø3Ø IF Z=F*8+33 AND U3<5 THEN CALL TREFFER(I,GR,TYP(),V(),
     T,TA,U3,VI,A)
6Ø4Ø NEXT I :: SUBEND
62ØØ SUB KURS(OO,K$):: CALL CLTAFEL
621Ø DISPLAY AT(2Ø,3)SIZE(9)::"dein`kurs"
622Ø ACCEPT AT(2Ø,15)VALIDATE("nsowØ")BEEP SIZE(-2):K$
     :: IF ASC(K$)=48 THEN SUBEXIT
623Ø OO1=OO :: FOR I=1 TO LEN(K$):: K1$=SEG$(K$,I,1)
     :: OO1=OO1+POS("s.w`o.n",K1$,1)-4
624Ø IF OO1>9  OR OO1<1 THEN 628Ø
625Ø IF (K1$="o" AND(OO1=4 OR OO1=7))OR(K1$="w" AND
     (OO1=6 OR OO1=3))THEN 628Ø
627Ø NEXT I :: OO=OO1 :: SUBEXIT
628Ø RESTORE 4Ø6Ø :: CALL TAFEL :: GOTO 621Ø :: SUBEND
64ØØ SUB TAFEL
     :: CALL CLTAFEL :: FOR I=1 TO 4 :: READ S$(I)
     :: DISPLAY AT(19+I,3)SIZE(LEN(S$(I))):S$(I)
     :: NEXT I :: SUBEND
65ØØ SUB MUSIK
     :: CALL ZEIT(2ØØ)
651Ø CALL SOUND(3ØØ,196,2,247,11):: CALL ZEIT(6Ø)
     :: FOR I=1 TO 2 :: CALL SOUND(8Ø,262,5,33Ø,7)
     :: CALL ZEIT(3Ø)
652Ø CALL SOUND(15Ø,262,5,33Ø,7):: CALL ZEIT(7Ø):: NEXT I
     :: CALL SOUND(15Ø,294,5,349,7):: CALL ZEIT(7Ø)
653Ø CALL SOUND(1ØØ,33Ø,5,392,7):: CALL ZEIT(3Ø)
     :: CALL SOUND(35Ø,262,5,33Ø,7):: SUBEND
```

*Listing zu »Madagaskar« (Schluß)*



```
# 1 Peilsprite
# 2...5 Peil-Himmelsrichtungen (Buchstaben)
# 4 Explosion
# 5 Explosionsblitz
# 6...9 Schiffe
# 10 Torpedo
# 15,16 Geländerabdeckung
# 27 »unsichtbarer« Sprite zum Abdecken sinkender Schiffe
# 28 untergehendes Schiff
```

***Sprites und ihre Bedeutung im Spiel »Madagaskar«***

Zieleinstellung und der Schußzahl der Salve abhängen. Bei Maschinenschaden verlangsamt sich die Geschwindigkeit des getroffenen Schiffes. Volltreffer führen zu seiner Explosion und zum Sinken des Schiffes.

Während eines Angriffs kann beliebig oft zwischen Tauchstation (Periskop) und aufgetauchtem Zustand (Kanone) gewechselt werden.

## Peilung mit Funk im aufgetauchten Zustand

Im aufgetauchten Zustand kann der Funkverkehr des Gegners angepeilt werden. Der Zeiger des Peilinstrumentes wird mit den Visiertasten »z« und »,« verstellt. Gegnerische Schiffe befinden sich in der Richtung, aus der der Funkverkehr am deutlichsten zu hören ist, ohne daß daraus auf ihre Entfernung geschlossen werden kann. Während das Gerät verstellt wird, ist kein Empfang möglich. Funkgeräusche, die nicht mehr auf die maximale Lautstärke einzustellen sind, stammen von Schiffen, die das Operationsgebiet verlassen haben und sich außerhalb des Aktionsradius des U-Bootes befinden.

Das Spiel ist zu Ende, wenn die Munition (12 Torpedos und 99 Schuß Kanonenmunition) aufgebraucht ist (Neustart mit »FCTN CLEAR« und »RUN«). Die abgeschossenen Schiffe werden in einer Leiste über den Instrumenten angezeigt. Ist die Reihe voll, so erhält der Spieler den Kabeljau-Orden. Es ist bei geschicktem Spiel möglich, drei Orden zu erreichen; ein Anfänger dürfte jedoch Mühe haben, die Reihe auch nur einmal voll zu bekommen.

## Hinweise für ein effizientes Spiel

☐ Nach dem Auftauchen muß gewartet werden, bis die Schrifttafel vollständig ausgedruckt ist, sonst »klemmen« die Bedienungselemente.

☐ Weit entfernte Schiffe bekämpft man günstiger mit der Kanone (Einzelschuß!), nähere mit Torpedo.

☐ Schiffe fahren in kleinen Gruppen, teilweise außer

**Speicherbelegung für Funkpeilung** 3000-3020 Umwandlung der Seegebietsziffern ORT(I) in kartesische Koordinaten X(I), Y(I)
3030-3040 Translation des Koordinatensystems: Umwandlung der Koordinaten in solche mit Zentrum 00 (Standort des U-Boots)
3050-3080 Berechnung der Winkel W gedachter Peilstrahlen von 00 zu den Standorden ORT(I) der Schiffe, Umrechnung auf Bogenlänge 16, mit Fallunterscheidungen zur Berücksichtigung der Hauptwerte der Tangensfunktion.
3090 Drei Elementen des Feldes LS( ) wird die maximale Lautstärke 4 zugeordnet
3100-3150 Belegung der Nachbarplätze mit den Lautstärken 15 und 26
**Periskop** erzeugt Bildschirmmaske für Ausblick im getauchten Zustand
**Schiff** erzeugt die Schiffssprites und ruft die Visiereinrichtungen (Kanone, Torpedo) auf
5120-5160 Auswahl der Zeilennummern (das heißt der »Entfernungen« der Schiffe), die auch die Geschwindigkeiten V(NR) der Schiffssprites beeinflussen.
5165-5200 Auswahl der Zeichen für die Sprites, abhängig von Zeile und Richtung der Bewegung.
**Kontakt** stellt fest, ob Schiffe und Torpedos am Rand angekommen sind und gelöscht werden müssen oder Treffer erfolgt sind.
**Torpedo** erzeugt Torpedosprites zweier Typen, je nach Sprite-Vergrößerungsfaktor GR. Die Torpedoreserve wird angezeigt.
**Treffer** löst die Sound- und Grafikeffekte im Falle eines Treffers aus, zusammen mit dem Unterprogramm »Erfolg«.
5510 Schiffssprite wird angehalten, Torpedosprite gelöscht.
5515, 5520 Schiffsprite wird umkopiert auf Sprite # 28, Explosionsblitz durch Sprite # 5, dessen Koordinaten X1, Y1 von der Art des Schusses abhängen (Schiffsmitte bei Treffer durch Kanone, Ort des Torpedos, Wasserlinie bei Treffer durch Torpedo).
5530-5570 Berechnung der Position eines »unsichtbaren« Sprites # 27, der unter das sinkende Schiff gelegt wird.
5590 Explosionsfontäne an der Stelle X2, Y2
5610 Aufruf des unsichtbaren Sprites; Schiffssprite wird hinter diesen gezogen
5620 Beide Sprites werden gelöscht
5527 Falls — bei Aktionsart »Kanone« — ein Schiff als teilweise beschädigt gilt (Indikator hierfür: U3), wird der kopierte Schiffssprite # 28 zurückverwandelt in einen Sprite mit der ursprünglichen Nummer, mit halber Geschwindigkeit.
5640-5657 Anzeige des abgeschossenen Schiffstyps; Kontrolle, ob Reihe voll ist.
**Visier1** Steuerung für Torpedoabschuß
**Visier2** Steuerung für Schwenkoptik und Abschußwinkel der Kanone
5760-5790 Addition der Schwenkgeschwindigkeit V1 der Zieloptik zur Geschwindigkeit V(I) der Schiffssprites, solange Visiertaste gedrückt wird.
5800-5810 Bewegung des Fadenkreuzes der Kanone
**Tauch** erzeugt die Bildschirmgrafik im »getauchten« Zustandes. Beim ersten Mal (Indikator E) werden 28 Fisch-Sprites aufgerufen, später maximal acht.
**Halt** wartet auf Tastatureingabe und erzeugt einen BEEP
**CLTafel** löscht die Anzeigentafel
**Kanone** 5970, 5980 Kanone schießt, solange die Space-Taste gedrückt wird.
5990, 5995 Anzeige der Munitionsreserve
6020-6030 Entscheidung über erfolgten Treffer und dessen Wirkung: U3 ist um so kleiner, je geringer der Abstand ABS(S-129 + Korrekturfaktor für Schiffsgröße) zwischen Schiffssprite und Visierspalte ist, und je größer die Schußzahl U1 der abgefeuerten Salve ist. U3<5 bedeutet Treffer, U3=0 Volltreffer.
**Kurs** berechnet den neuen Standort des U-Boots nach Eingabe der Fahrtrichtung. Es ist zu beachten, daß der eingegebene String K$ immer von der Länge 2 ist, auch dann, wenn nur eine einstellige Eingabe, zum Beispiel »n« erfolgt. K$ hat dann die Gestalt »nn«.
**Tafel** formatiert Texte und zeigt sie im dafür reservierten Anzeigenfeld.
**Musik** spielt das Lied »Wir lagen vor Madagaskar«

***Erläuterungen zu den Unterprogrammen***

Sichtweite des U-Boots. Um alle im Seegebiet aufzuspüren, sollte bei Pausen getaucht und ein kurzes Stück (Kurseingabe 0) gefahren werden.

Nach Ankunft im Zielgebiet ist eine Taste zu drükken, damit das Menü für Auftauchen oder Weiterfahrt erscheint.

Das Listing entspricht übrigens nicht ganz der tatsächlichen Eingabe am Bildschirm. Die Kommentare und Zwischenüberschriften wurden deswegen nicht als REM-Zeilen in das Programm eingebaut, weil sonst der Speicherplatz der Konsole nicht ausgereicht hätte (während des Spiels bleiben sogar ohne REM-Zeilen nur noch knappe 100 Bytes übrig). Diese Anmerkungen im Listing dürfen also nicht mit eingetippt werden. Die Leerräume in den Zeilen 39 bis 47 sind ebenfalls nur zur besseren Übersichtlichkeit im Listing eingefügt. Bis auf den Leerraum nach »DATA« sind sie wegzulassen. (Dieter Taube)

| ZS | Farbe | Zeichen |
|---|---|---|
| 1 | dunkelblau/dunkelgrün auf schwarz | 36 - 39<br>34,35:Fische |
| 2 | hellblau auf hellgrün | 44 41 - 43 * 40 |
| 3<br>4 | dunkelgrün auf hellgrün | ZAHLEN<br>58 59 60 - 63:Explosion |
| 5 | grau/gelb/hellrot | 64 65 66<br>67 68 70<br>69: Schiff |
| 6<br>7 | schwarz auf hellgrün | GROßBUCHSTABEN H ... W<br>P = A, Q = E<br>+ 86 |
| 8 | hellrot | 88 89 93 - 95 94,92 Torpedo |
| 9<br>10<br>11 | schwarz auf dunkelgrün | KLEINBUCHSTABEN bis w<br>96 |
| 12 | schwarz auf dunkelgrün | x,y,z<br>125 = ö, 124 = ä |
| 13 | gelb | 128-131 Geländer 132 - 135 Schiffe |
| 14 | gelb | 136 - 143: Schiffe |

***Zuordnung der Zeichensätze nach Farben***

# TI-Basic um »DISPLAY AT« und »ACCEPT AT« erweitert

**Fast jeder TI 99/4A-Besitzer kennt die leistungsfähigen Ein- und Ausgabebefehle des Extended Basic für den Bildschirm. Viele bedauern, daß diese Befehle in TI-Basic nicht zur Verfügung stehen. Dem kann abgeholfen werden!**

TI 99/4A

Wer ein Mini Memory Modul (oder Editor/Assembler Modul) besitzt, der wird das TMS-9900-Assembler-Programm »@DSPACPTS« (Listing 1) bald zu schatzen wissen. Es erweitert TI-Basic um die Befehle: »CALL LINK (»DSPLAY«, Zeile, Spalte, Länge, Ausgabestring)« und »CALL LINK (»ACCEPT«, Zeile, Spalte, Länge, Ausgabestring)«. Die Befehle sind nicht ganz so umfangreich wie die von Extended Basic her bekannten Gegenstücke. So sind zum Beispiel für Ein- und Ausgabe die Variablen auf Stringvariable oder Stringausdrücke beschränkt. Die Optionen (außer »SIZE«) fehlen völlig. Dennoch sind die Befehle sehr leistungsstark, und alle von Basic her bekannten Editiermöglichkeiten wurden realisiert.

## Bildschirmadressierung des TI 99/4A

Den Inhalt des Bildschirmes speichert der TI 99/4A im VDP-RAM, in Basic und Extended Basic beginnend mit Adresse >0000. Da wir hier nicht mit der Verwendung von Farbe konfrontiert sind, interessieren uns jetzt nur zwei Speicherbereiche des VDP-RAMs.
Screen Image Table:
>0000 bis >0300 hexadezimal
0 bis 767 dezimal
Pattern Descriptor Table:
>0000 bis >05FF hexadezimal
0 bis 1536 dezimal
Jedes Byte des »SCREEN IMAGE TABLE« korrespondiert mit einer Bildschirmposition eines Characters. VDP-RAM-Adressen >0000 bis >001F entsprechen der ersten Bildschirmzeile, VDP-RAM-Adressen >0070 bis >003F der zweiten usw.

Da jede Character-Nummer in einem Byte enthalten ist, kann man erkennen, daß Character-Nummern zwischen >00 und >FF hexadezimal oder 0 und 255 dezimal liegen können. Die »PATTERN DESCRIPTOR TABLE« ist der zweite Speicherbereich, den wir benützen. Dieser Bereich enthält die Dotmuster für jedes der 256 Zeichen, die wir verwenden können. Das Basic-Subprogramm »CHAR«, welches dazu verwendet wird, Sonderzeichen zu definieren, speichert die Muster in dieser Tabelle. In Basic und Extended Basic beginnt diese Table im VDP-RAM ebenfalls mit Adresse >0000, also identisch dem »SCREEN IMAGE TABLE«.

Wie wir später sehen werden, müssen wir deshalb alle ASCII-Codes für die interne Verwendung im TI 99/4A transformieren. Um ein Zeichen auf dem Bildschirm darzustellen, muß zu seinem ASCII-Code 96 hinzugezählt werden (zum Beispiel: Cursor ASCII-Code 30 → intern 30 + 96 = 126).

Für eine bestimmte Bildschirmzeile und -spalte wird die Bildschirmposition im VDP-RAM folgendermaßen berechnet: »(ZEILE - 1) x 32 + SPALTE - 1«. Schreibt man eine Character-Nummer, zum Beispiel »@« → 64 + 96 = 160« in die Adresse >0000 des VDP-RAMs, so erscheint an Position (0,0) das Zeichen »@«. Den Offset für den »PATTERN DESCRIPTOR TABLE« (CHAR - 1) x 8, hat der Computer automatisch berechnet.

Wenn der »HOME«-Character auf dem Bildschirm ein »@« anzeigt, so enthält die VDP-RAM-Adresse 0 den Wert >A0 oder 160.

```
100 REM DISPLAY
110 REM *******
120 REM
130 REM BY @APESOFT
140 REM
150 CALL CLEAR
160 M$="@APESOFT"
170 Z=1
180 S=1
190 GOSUB 260
200 GOTO 200
210 REM
220 REM
230 REM DISPLAY AT
240 REM **********
250 REM
260 OFFSET=(Z-1)*8+1
270 FOR I=1 TO LEN(M$)
280 CALL LOAD(OFFSET+I,
    ASC(SEG$(M$,I,1))+96)
290 NEXT I
300 RETURN
```

*Listing 2 Direktes Display (über LOAD)*

Der Offset zur »PATTERN DESCRIPTOR TABLE« ist dann 160 x 8 = 1280 oder >0500. Die 8 Bytes, beginnend mit Adresse >0500, enthalten den Musterstring für »@«. Diese 8 Bytes entsprechen ganz genau der zeilenweisen Darstellung des Characters durch den TI 99/4A.

Welche Konsequenzen diese Anordnung der »SCREEN IMAGE« und »PATTERN DESCRIPTOR« Tabellen hat, konnte jeder feststellen, der versuchte, direkt Daten in die »SCREEN IMAGE« Tabelle ►

```
*     UNL
*
*       **********************
*      ************************
*     **************************
*     ***                    ***
*     ***                    ***
*     ***   T I - 9 9 / 4 A  ***
*     ***                    ***
*     ***                    ***
*     ***     DISPLAY  AT     ***
*     ***                    ***
*     ***         und        ***
*     ***                    ***
*     ***      ACCEPT AT     ***
*     ***                    ***
*     ***                    ***
*     ***          by        ***
*     ***                    ***
*     ***   @ A P E S O F T  ***
*     ***                    ***
*     ***                    ***
*      **************************
*       ************************
*        **********************
*
*     TMS 9900  ASSEMBLER  Version 1.1 (MINI MEMORY MODUL)
*
*     Programmname: "@DSPACPTS ("@DSPACPTS")
*
*     ENZERSDORF  1983-10-16
*
*     Dieses Programm erweitert TI BASIC mit den von EXTENDED BASIC
*     her bekannten Routinen: "DISPLAY AT" und "ACCEPT AT".
*     Diese Befehle stehen somit fuer MINI MEMORY  oder
*     EDITOR ASSEMBLER Modul zur Verfuegung.
*
*     Die Befehle lauten:
*
*     CALL LINK("DSPLAY",ROW,COL,LEN,VAR$)
*     CALL LINK("ACCEPT",ROW,COL,LEN,VAR$)
*
*
*     Diese Befehle werden bei den einzelnen Programmsektionen
*     genauer erlaeutert.
```

*Listing Display/ Accept*

```
       IDT  DISPACPT
*
*      ******************
*      * DEKLARATIONEN *
*      ******************
*
*      REFERENZEN fuer EDITOR/ASSEMBLER Modul
*
*                                Bei der Herstellung eines
*      REF  NUMREF,STRREF        Objectcodes fuer E/A Modul
*      REF  STRASG,XMLLNK,ERR    muessen die folgenden EQUATES
*                                entfernt werden.
       AORG >7A00                Wir generieren absoluten Code
*
       DEF DSPLAY,ACCEPT
*
*
*      EQUATES fuer MINI MEMORY Modul
*
*
NUMREF EQU  >0044        Uebernimmt numerische Parameter
STRREF EQU  >004C        Uebernimmt String Parameter
STRASG EQU  >0048        Uebergibt String Parameter
XMLLNK EQU  >001C        Interne Maschinensprache Routine
ERR    EQU  >0050        Fehler Report
*
*      Sonstige EQUATES
*
CFI    EQU  >1200        Real --> Integer Transformation
ERRBA  EQU  >1600        Bad Argument
*
SCRTBL EQU  >0000        SCREEN IMAGE TABLE
PTDTBL EQU  >0000        PATTERN DESCRIPTOR TABLE (BASIC)
KEY    EQU  >8375        ASCII Wert der gedrueckten Taste
*
       EVEN
*
MYWS   BSS  32           Eigene Arbeitsregister
R0HB   EQU  MYWS         Absolute Adressen fuer
R0LB   EQU  MYWS+1       Arbeitsregister R0
*
FAC    EQU  >834A        FPOINT Akkumulator
STATUS EQU  >837C        GPL Status Byte
GPLWS  EQU  >83E0        GPL Arbeitsregister
*
*      KONSTANTE, VARIABLE
*
PRMTR  BSS  2            LINK Parameter Buffer
ROW    BSS  2            Zeile
COL    BSS  2            Spalte
*
H00    DATA 0            Wortkonstante
H29    DATA 29
H30    DATA 30
*
RTLEV1 BSS  2            Buffer fuer Returnadressen in
RTLEV2 BSS  2            verschiedenen Leveln
RTLEV3 BSS  2
*
STRING BSS  33           Buffer fuer String
*
BYT1   BYTE 1            Byte Konstante
BYT28  BYTE 28
BYT96  BYTE 96
BYTM96 BYTE -96
*
CLEAR  BYTE 2            Taste CLEAR
ERASE  BYTE 7            Taste ERASE
LEFT   BYTE 8            Pfeil  <--
RIGHT  BYTE 9            Pfeil  -->
ENTER  BYTE 13           Taste ENTER
NOKEY  BYTE >FF          keine Taste gedrueckt
CURSOR BYTE 126          CURSOR Character (BASIC)
SCRCHR BSS  1            Buffer fuer SCREEN Character
*
       EVEN
*
*
       COPY "DSK2.@DSPACPT1S"    Subprograme
       COPY "DSK2.@DSPACPT2S"    Hauptprograme
*
       END
       Segment "@DSPACPT1S"
*
*       *************************
*      ***************************
*      **                       **
*      **                       **
*      **  VDP-ACCESSROUTINEN   **
*      **                       **
*      **                       **
*      ***************************
*       *************************
*
*
*      Diese Routinen werden anstelle der roamresidenten  Routinen
*      verwendet, um fuer EDITOR/ASSEMBLER und MINI MEMORY Modul
*      einen gemeinsamen Code herzustellen.
*
*      ACHTUNG:     Diese Routinen verwenden  Register
*      --------     R13 als Zwischenbuffer!
*
*
*
*      ************************
*      *  VSBW  SUBROUTINE   *
*      ************************
*
*
*      FORMAT:      BL   @VSBW
*
*      Register:    R0...VDP-Adresse
*                   R1...zu schreibendes Byte (MSB)
*
*
VSBW   MOV  R0,R13           VDP-RAM Adresse sichern
       ORI  R0,>4000
       MOVB @R0LB,@>8C02     VDP-WRITE-DATA Register
       MOVB @R0HB,@>8C02     setzen
       MOVB R1,@>8C00        R1 MSByte ins VDP-RAM schreiben
       MOV  R13,R0
       RT


*      ************************
*      *  VSBR  SUBROUTINE   *
*      ************************
*
*
*      FORMAT:      BL   @VSBR
*
*      Register:    R0...VDP-Adresse
*                   R1...gelesenes Byte (MSB)
*
*
VSBR   SWPB R0
       MOVB R0,@>8C02    VDP-READ-DATA Register
       SWPB R0           setzen.
       MOVB R0,@>8C02
       MOVB @>0000,R1    Byte aus VDP-RAM nach MSByte von R1
       RT
*
*
*
*      *********************
*      * KSCAN SUBROUTINE *
*      *********************


*      FORMAT:   BL   @KSCAN
*
*      Fragt die Tastatur ab und liefert die Variablen
*      KEY und STATUS entsprechend der Eingabe zurueck.
*
*
KSCAN  MOVB @H00,@>8374  Checkt die gesamte Tastatur
       LWPI GPLWS        GPL Workspace laden
       MOV  R11,@RTLEV3
       BL   @>000E       SCAN Konsolen Adresse
       MOV  @RTLEV3,R11
       LWPI MYWS         Eigene Arbeitsregister laden
       RT


*       ******************************
*      ********************************
*      **                            **
*      **  DIVERSE UNTERPROGRAMME    **
*      **                            **
*      ********************************
*       ******************************
*
*
*      RUECKKEHR INS BASICPROGRAMM
*      ***************************
*
*      FORMAT:  BL   @RTN
*
*
RTN    MOVB @H00,@STATUS      GPL-STATUS Byte loeschen
       LWPI GPLWS             GPL Arbeitsregister laden
       B    @>0070            zurueck zu Basic
*
*
*      SUBPROGRAM  BAD ARGUMENT
*      ************************
*
*      FORMAT:  BL  @BADARG
*
*
BADARG LI   R0,ERRBA     Fehlercode nach MSByte von R0
       BLWP @ERR         "BAD ARGUMENT" Anzeige und Stop
```

*Listing Display/ Accept (Fortsetzung)*

```
*       SUBPROGRAMM LINK PARAMETER HOLEN
*       ********************************
*
*       FORMAT:   BL  @GETPAR
*
*
GETPAR  BLWP @NUMREF            Numerischen Parameter uebernehmen
        BLWP @XMLLNK            ihn in Integer verwandeln
        DATA CFI
        RT
*
*
*       SUBPROGRAMM  ZEILEN/SPALTEN PARAMETER UEBERNEHMEN
*       *************************************************
*
*
*       FORMAT              BL     @GETPOS
*                           DATA   UGRENZE,OGRENZE,ZIEL
*
*
*       Variable:  PRMTR...Link Parameter
*
*       Uebernimmt Zeilen- und Spaltenparameter und ueberprueft
*       ob sie mit den angegebenen Grenzen uebereinstimmen.
*
*
GETPOS  MOV  *R11+,R2       Untere Grenze
        MOV  *R11+,R3       Obere Grenze
        MOV  *R11+,R4       Zieladresse
        MOV  R11,@RTLEV1    Returnadresse sichern
*
        DEC  R2             Grenzen fuer Vergleich
        INC  R3             aufbereiten
*
        CLR  R0             Link Parameter Nummer
        MOV  @PRMTR,R1      holen
        BL   @GETPAR        Wert holen und in
        MOV  @FAC,*R4       ZIEL speichern
*
        C    *R4,R2         Untere Grenze testen
        JGT  TSTOG          Okay
        B    @BADARG        Grenze unterschritten
TSTOG   C    *R4,R3         Obere Grenze testen
        JLT  GTPSRT         Okay
        B    @BADARG        Grenze ueberschritten
*
GTPSRT  MOV  @RTLEV1,R11    Returnadresse gewinnen
        RT


*       SUBPROGRAMM UEBERNAHME ROW, COL UND SIZE
*       ****************************************
*
*       FORMAT:           BL  @GETRCS
*
*       Dieses Segment uebernimmt Anzeigeposition und
*       Laenge eines Strings. Ist die Stringlaenge
*       SIZE positiv, werden SIZE Character an der
*       Anzeigeposition am Bildschirm geloescht.
*
SIZE    EQU  9              SIZE ist immer in R9
LEN     EQU  10             LEN ist immer in R10
*
GETRCS  MOV  R11,@RTLEV2    Returnadresse sichern
        LI   R1,1           ROW uebernehmen
        MOV  R1,@PRMTR      Link Parameter 1 fuer ROW
        BL   @GETPOS        Grenzen  1 <--> 24
        DATA 1,24,ROW
*
        INC  @PRMTR         COL uebernehmen, Link Param. 2
        BL   @GETPOS        Grenzen  1 <--> 28
        DATA 1,28,COL       1. Druckposition ist
        INC  @COL           Bildschirmspalte 3!
*
        INC  @PRMTR         SIZE uebernehmen, Link Param. 3
        BL   @GETPOS        Grenzen  -28 <--> +28
        DATA -28,28,MYWS+18
*
        DEC  @ROW
        MOV  @ROW,R3        Zeilenposition in Screen Image Table
        SLA  R3,5           (ROW-1)
*
        MOV  SIZE,SIZE      SIZE negativ?
        JEQ  RCSRT2         0 Positionen --> beenden
        JLT  RCSRT1         SIZE ist negativ
        BL   @DSIZE         SIZE Druckpositionen loeschen
RCSRT1  ABS  SIZE
        MOV  @RTLEV2,R11    Returnadresse gewinnen
        RT
*
RCSRT2  B    @RTN           Bei Nullstring retour zu Basic
*
*
*
*       SUBPROGRAMM DRUCKPOSITIONEN LOESCHEN
*       ************************************
*
*       FORMAT:           BL  @DSIZE
*
*       Loescht beginnend mit Position ROW und COL
*       SIZE Character.
*
DSIZE   MOV  R11,@RTLEV3    Returnadresse sichern
        MOV  @COL,R4
        MOV  SIZE,R5
        LI   R1,>0000       Leerzeichen (Basic 128!)
DSIELP  MOV  R3,R0          SCRTBL Offset berechnen
        A    R4,R0          SCRTBL OFFSET ist nun in R0
        BL   @VSBW          Geloescht wird Byte fuer
        DEC  R5             Byte
        JEQ  DSIERT
        INC  R4
        C    R4,@H30        letzte Spaltenposition erreicht?
        JLT  DSIELP         Nein!
*
DSIERT  MOV  @RTLEV3,R11    Returnadresse wiedergewinnen
        RT
*
*
*
*       SUBPROGRAMM TASTATUR ABFRAGEN
*       *****************************
*
*       FORMAT:      BL  @KINPUT
*
*       Dieses Modul checkt die gesamte Tatstatur und blinkt
*       waehrend der Warteposition abwechselnd mit dem CURSOR
*       und dem gegenwaertig an der SCRTBL Position
*       befindlichen Character.
*
KINPUT  MOV  R11,@RTLEV1 Returnadresse sichern
        BL   @VSBR       momentanen SCREEN Character
        MOVB R1,@SCRCHR  in Buffer schreiben
*
KEYLP1  LI   R15,200     In der ersten Abfrageschleife
        MOVB @CURSOR,R1  CURSOR anzeigen
        BL   @VSBW
KEYLP2  BL   @KSCAN      Tastatur checken
        CB   @KEY,@NOKEY Keine Taste gedrueckt?
        JNE  KEYRT       doch --> return
        DEC  R15         Naechster Check
        JOC  KEYLP2
*
        LI   R15,250     In der zweiten Abfrageschleife
        MOVB @SCRCHR,R1  gegenwaertigen SCREEN Character anzeigen
        BL   @VSBW
KEYLP3  BL   @KSCAN      Tastatur checken
        CB   @KEY,@NOKEY keine Taste gedrueckt?
        JNE  KEYRT       doch --> return
        DEC  R15         naechster Check
        JOC  KEYLP3
        JMP  KEYLP1      Bisher kein Erfolg
*
KEYRT   MOVB @SCRCHR,R1  gegenwaertigen SCREEN Character
        BL   @VSBW       wieder anzeigen (damit nicht ploetzlich
        MOV  @RTLEV1,R11 z.B. der CURSOR bei ENTER haengt!)
        RT
*
*
*
*       SUBPROGRAMM WARTESCHLEIFE
*       -------------------------
*
*       FORMAT:     BL    @DELAY
*                   DATA  TIME
*
*       Schaltet eine Pause ein
*
DELAY   MOV   *R11+,R15    Zaehler fuer  Warteschleife
DLYLP   NOP                Tut nichts
        DEC   R15          Zaehler erniedrigen
        JOC   DLYLP
        RT
*
*
*       Ende "@DSPACPT1B"
*        Programmsegment "@DSPACPT2B"

*
*        ********************
*       **********************
*       **                  **
*       **  D I S P L A Y   **
*       **                  **
*       **********************
*        ********************
*
*
*
*       CALL LINK("DSPLAY",ROW,COL,SIZE,VAR$)
*       *************************************
*
*
*
*       Diese Routine entspricht dem DISPLAY AT von EXTENDED
*       BASIC, jedoch ohne Optionen.
*
*       ROW.......Zeilendruckposition   (1-24)
*       COL.......Spaltendruckposition  (1-28)
```

**Listing Display/ Accept (Fortsetzung)**

```
*      SIZE.....max. 28 Zeichen der Variablen
*              +SIZE Positionen werden vor Anzeige geloescht
*              -SIZE Positionen werden vor Anzeige nicht geloescht
*      VAR$.....Stringvariable oder Stringausdruck
*
*
DSPLAY LWPI MYWS        Eigene Arbeitsregister laden
       BL   @GETRCS     ROW,COL,SIZE uebernehmen
*
*      STRING UEBERNEHMEN
*
       CLR  R0
       LI   R1,4        String ist 4. Parameter in LINK
       LI   R2,STRING   STRING darf maximal
       MOVB @BYT28,*R2  28 Bytes lange sein!
       BLWP @STRREF     String uebernehmen
       MOVB @STRING,LEN Aktuelle Stringlaenge gewinnen
       SRL  LEN,8       Byte --> Wort
       JEQ  DSPRT       Nullstring --> beenden
*
*      STRING AM BILDSCHIRM AUSGEBEN
*
       LI   R2,STRING+1 STRINGADRESSE
DSPYLP MOV  R3,R0
       A    @COL,R0     SCRTBL OFFSET
       MOVB *R2+,R1     zu jedem ASCII Code muss 96 hinzugezaehlt wer
den,
       AB   @BYT96,R1   um zum BASIC PTDTBL Offset zu gelangen
       BL   @VSBW
*
       DEC  SIZE        Alle Positionen gedruckt?
       JEQ  DSPRT       ja
       DEC  LEN         STRING zu Ende?
       JEQ  DSPRT       ja
       INC  @COL        Groesste Spaltenposition ist 30
       C    @COL,@H30
       JLT  DSPYLP
*
DSPRT  B    @RTN


*       ******************
*      ********************
*      **                **
*      **  A C C E P T   **
*      **                **
*      ********************
*       ******************


*      CALL LINK("ACCEPT",ROW,COL,SIZE,VAR$)
*      *************************************


*      Diese Routine entspricht dem ACCEPT AT von EXTENDED
*      BASIC, jedoch nur mit beschraenkten Optionen.
*
*      ROW.......Zeileneingabeposition   (1-24)
*      COL.......Spalteneingabeposition  (1-28)
*      SIZE......max. 28 Zeichen der Variablen
*              +SIZE Positionen werden vor Anzeige geloescht
*              -SIZE Positionen werden vor Uebernahme geloescht
*      VAR$.....Stringvariable
*
*
*      Register:   R0,R1,R2...VDP-Acces
*                  R3.........Offset aus Eingabezeile
*                  R6.........Buffer fuer ROW
*                  R7.........Buffer fuer SIZE



ACCEPT LWPI MYWS        Eigene Arbeitsregister laden
       BL   @GETRCS     ROW,COL UND SIZE uebernehmen
*
*      EINGABE DER CHARACTER
*
COACPT MOV  @COL,R6
       LI   R7,1
ACPTLP BL   @DELAY      Eingabeverzoegerung
       DATA 8888
       MOV  R3,R0       SCRTBL Offset
       A    R6,R0       ist in R0
       BL   @KINPUT     Tastatur abfragen
*
       CB   @KEY,@LEFT  Taste "<--" gedrueckt?
       JNE  TRIGHT      nein
       C    R6,@COL     linker Anschlag erreicht?
       JEQ  ACPTLP      Ja.....Position halten
       DEC  R6          Nein...Position des Cursors erniedrigen
       DEC  R7
       JMP  ACPTLP
*
TRIGHT CB   @KEY,@RIGHT Taste "-->" gedrueckt?
       JNE  TERASE      nein
       C    R6,@H29     rechter Anschlag erreicht?
       JEQ  ACPTLP      Ja.....Position halten
       C    R7,SIZE     Buchstabenanzahl erreicht?
       JEQ  ACPTLP      Ja.....Position halten
       INC  R6          Nein...Position des Cursors erniedriger.
       INC  R7
       JMP  ACPTLP
*
TERASE CB   @KEY,@ERASE Taste "ERASE" gedrueckt?
       JNE  TCLEAR      nein
       BL   @DSIZE      Eingabepositionen wieder loeschen
       JMP  COACPT
*
TCLEAR CB   @KEY,@CLEAR Taste "CLEAR" gedrueckt?
       JNE  TENTER      nein
       B    @BADARG     Programm stoppt mit
*                       "BAD ARGUMENT" Anzeige
*
TENTER CB   @KEY,@ENTER Taste "ENTER" gedrueckt?
       JEQ  GETSTR      Ja.....Inputstring uebernehmen
*
       AB   @BYT96,@KEY SREEN Character fuer Basic
       MOVB @KEY,R1
       BL   @VSBW       Character anzeigen
*
       C    R6,@H29     rechter Rand erreicht?
       JEQ  ACPTLP      ja
       C    R7,SIZE     Characteranzahl erreicht?
       JEQ  ACPTLP      ja
       INC  R6          Naechste Spalte
       INC  R7          Naechster Character
       JMP  ACPTLP
*
*      UEBERNAHME DER CHARACTER VOM SCREEN
*
GETSTR LI   R2,STRING   CPU-Adresse des Strings
       MOVB @H00,*R2+   Characteranzahl loeschen
GSTRLP MOV  R3,R0       SCRTBL Offset berechnen
       A    @COL,R0
       BL   @VSBR       Character lesen
       AB   @BYTM96,R1  in ASCII Format transformieren
       MOVB R1,*R2+     und in String schreiben
       AB   @BYT1,@STRING Characteranzahl erhoehen
       DEC  SIZE        Naechstes Byte
       JEQ  ACPTRT
       INC  @COL        Naechste Spalte
       C    @COL,@H30   letze Eingabespalte erreicht?
       JLT  GSTRLP
*
*      STRING ZUWEISEN
*
ACPTRT MOVB @H00,@STATUS  GPL STATUS Byte loeschen

       CLR  R0
       LI   R1,4         Link Parameter 4 fuer String
       LI   R2,STRING    STRING an Basicvariable
       BLWP @STRASG      zuweisen
*
       B    @RTN         naechste Basicanweisung
*
*
*      ENDE "@DSPACPT2S"
```

*Listing Display/Accept (Schluß)*

**Pokeliste**

```
100 REM DISPLAY/ACCEPT POKE LISTE
110 REM ************************
120 REM
130 by @APESOFT
140 REM
150 OPTION BASE(1)
160 DIM B(16)
170 A=31232
180 REM
190 REM DATEN FUER TMS9900 OBJECT KODE
200 REM ==============================
210 DATA 22,0,145,3,255,227,0,32,122,18,0,0,0,4,0,0
220 DATA 0,0,0,10,0,0,124,60,0,0,0,36,0,0,0,0
230 DATA 0,3,0,1,0,4,0,0,0,29,0,30,124,60,124,0
240 DATA 96,116,0,49,49,49,49,32,32,32,32,32,32,0,0,0
250 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
260 DATA 0,0,0,1,28,96,160,2,7,0,9,13,255,126,145,0
270 DATA 195,64,2,96,64,0,216,32,122,1,140,2,216,32,122,0
280 DATA 140,2,216,1,140,0,192,13,4,91,6,192,216,0,140,0
290 DATA 6,192,216,0,140,2,200,96,136,0,4,91,216,32,122,0
300 DATA 131,116,2,224,131,224,200,11,122,48,6,160,0,14,194,0
310 DATA 122,48,2,224,122,0,4,91,216,32,122,38,131,124,2,0
320 DATA 131,224,4,96,0,112,2,0,22,0,4,32,96,80,4,0
330 DATA 96,60,4,32,96,28,10,0,4,91,192,187,192,251,193,0
340 DATA 200,11,122,44,6,2,5,131,4,192,192,96,122,32,6,0
350 DATA 122,198,197,32,131,74,128,140,21,2,4,96,122,182,128,0
360 DATA 17,2,4,96,122,182,194,224,122,44,4,91,200,11,122,0
```

```
370 DATA 2,1,0,1,200,1,122,32,6,160,122,202,0,1,0,0
380 DATA 122,34,5,160,122,32,6,160,122,202,0,1,0,28,122,0
390 DATA 5,160,122,36,5,160,122,32,6,160,122,202,255,228,0,0
400 DATA 122,18,6,32,122,34,192,224,122,34,10,83,194,73,19,0
410 DATA 17,2,6,160,123,82,7,73,194,224,122,46,4,91,4,0
420 DATA 122,160,200,11,122,48,193,32,122,36,193,73,2,1,128,0
430 DATA 192,3,160,4,6,160,122,96,6,5,19,4,5,132,136,0
440 DATA 122,42,17,246,194,224,122,48,4,91,200,11,122,44,6,0
450 DATA 122,122,216,1,122,94,2,15,0,200,200,96,122,93,6,0
460 DATA 122,96,6,160,122,140,152,32,131,117,122,92,22,17,6,0
470 DATA 24,248,2,15,0,250,200,96,122,94,6,160,122,96,6,0
480 DATA 122,140,152,32,131,117,122,92,22,3,6,15,24,248,16,0
490 DATA 200,96,122,94,6,160,122,96,194,224,122,44,4,91,195,0
500 DATA 16,0,6,15,24,253,4,91,2,224,122,0,6,160,122,0
510 DATA 4,192,2,1,0,4,2,2,122,50,212,160,122,84,4,0
520 DATA 96,76,210,160,122,50,9,138,19,28,2,2,122,51,192,0
530 DATA 160,32,122,36,200,114,176,96,122,85,6,160,122,96,6,0
540 DATA 19,0,6,10,19,6,5,160,122,36,136,32,122,36,122,0
550 DATA 17,238,4,96,122,160,2,224,122,0,6,160,122,252,193,0
560 DATA 122,36,2,7,0,1,6,160,123,206,31,64,192,3,160,0
570 DATA 6,160,123,122,152,32,131,117,122,89,22,6,136,6,122,0
580 DATA 19,242,6,6,6,7,16,239,152,32,131,117,122,98,22,0
590 DATA 136,6,122,40,19,232,138,71,19,238,5,134,5,135,16,0
600 DATA 152,32,131,117,122,88,22,3,6,160,123,82,16,216,152,0
610 DATA 131,117,122,87,22,2,4,96,122,182,152,32,131,117,122,0
620 DATA 19,15,184,32,122,85,131,117,200,94,131,117,6,148,122,0
630 DATA 136,6,122,40,19,200,138,71,19,198,5,134,5,135,16,0
640 DATA 2,2,122,50,220,160,122,38,192,3,160,32,122,36,6,0
650 DATA 122,122,176,96,122,86,228,129,184,32,122,83,122,50,6,0
660 DATA 19,6,5,160,122,36,136,32,122,36,122,42,17,237,216,0
670 DATA 122,38,131,124,4,192,2,1,0,4,2,2,122,50,4,0
680 DATA 96,72,4,96,122,168,0,0,0,0,0,0,0,0,0,0
690 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
700 REM
710 REM OBJECT CODE IN MINI MEMORY POKEN
720 REM ================================
730 REM
740 FOR A=A TO A+768 STEP 16
750 FOR I=1 TO 16
760 READ B(I)
770 NEXT I
780 CALL LOAD(A,B(1),B(2),B(3),B(4),B(5),B(6),B(7),B(8),B(9),B(10),
B(11),B(12),B(13),B(14),B(15),B(16))
790 NEXT A
800 REM
810 REM DEF TABLE SETZEN
820 REM ================
830 REM
840 A=32767-16
850 I=0
860 M$="DSPLAY"&CHR$(123)&CHR$(216)&"ACCEPT"&CHR$(124)
&CHR$(38)
870 FOR A=A TO A+LEN(M$)
880 I=I+1
890 CALL LOAD(A,ASC(SEG$(M$,I,1)))
900 NEXT A
880 END
```

**Pokeliste (Schluß)**

zu poken. Zum Beispiel: >CALL LOAD (0,64) führt leider nicht dazu, daß ein »@« an der »HOME«-Position des Bildschirms erscheint. Zählt man allerdings ein Offset von 96 zum gewählten Byte hinzu, ist die Welt wieder in Ordnung.

>CALL LOAD (0,64+96) bringt nun tatsächlich »@« an die Position (1,1) des Bildschirms! Will man direkt Daten am Bildschirm anzeigen, kann die Subroutine aus Listing 2 sehr nützlich sein.

## Programm

Wenden wir uns nun der Erstellung des TMS 9900-Quellenprogrammes für »DISPLAY AT« und »ACCEPT AT« zu. Alle Erläuterungen befinden sich unmittelbar als REMarks im Quellencode. Da das Programm für eine direkte Eingabe über den »Line By Line Assembler« zu lang ist, haben wir mit Listing 3 ein Basic-Programm (DISPLAY AT/ACCEPT AT, Poke Liste) beigefügt. Aufgrund des ADRG >7A00-Statements zu Beginn des Quellencodes beginnt der Objektcode im Mini Memory-RAM mit Adresse 31232.

Das Programm in Listing 3 pokt den Objektcode direkt ins RAM und setzt die DEF Tabelle für »DSPLAY« und »ACCEPT« auf die letzten 16 Bytes des Mini Memory-RAM. (Dr. A. Peschetz)

Quelle: Dr. A. Peschetz, »99-Spezial/II«, Texas Instruments (in Vorbereitung). Nachdruck mit freundlicher Genehmigung des Autors.


# Listing des Monats

Programmieren Sie Ihren Computer selbst?
Haben Sie Programme, die Sie selbst geschrieben haben?
Wozu setzen Sie diese Programme ein?

Wir suchen die schönsten Listings unserer Leser, um sie in den nächsten Ausgaben zu veröffentlichen
Denn Happy-Computer ist die Zeitschrift zum Mitmachen.
Für jedes Listing, das in Happy-Computer erscheint, zahlen wir ein Honorar von DM 100,— bis zu DM 300,—.

## Bis zu DM 2.000,— zu gewinnen:

Die Redaktion von Happy-Computer prüft alle Einsendungen.
Aus den schönsten Listings, die veröffentlicht werden, wird einmal im Monat das »Listing des Monats« ausgesucht und mit einem Barbetrag von

**DM 2.000,—**
prämiert

Mitmachen Gewinnen

Jeden Monat Superchance

## Und so machen Sie mit:

Schicken Sie Ihr Listing und das ablauffähige Programm auf einem geeigneten Datenträger mit ausführlicher Beschreibung darüber, was Sie mit diesem Programm alles machen, wie es funktioniert und wie es aufgebaut ist (Flußdiagramm). Dazu eine Liste der Variablen mit möglichst vielen aussagefähigen Beispielen. Verwenden Sie für Ausdrucke und Listings ein neues Farbband und weißes Papier. Schicken Sie nur Originale — keine Kopien!

Schicken Sie Ihr Listing an:
Happy-Computer — Aktion:
Listing des Monats —
Hans-Pinsel-Str. 2,
8013 Haar bei München

dem Bildschirm (4 oder 8), die maximale Anzahl der Angreifer und die Punkte pro abgeschossenem Feind eingegeben werden. Die Bewegungsmuster je Angriffswelle werden mit Hilfe der Option 5) definiert. Hierbei wird in Form von Zahlenketten die Richtungsänderung der Aliens angegeben, die auf Wunsch auch Bomben werfen. Eine Linie zeigt die aktuelle Bewegung an und veranschaulicht so den Zahlencode. Die Handhabung dieser Option ist dadurch relativ einfach und unkompliziert.

Der Benutzer kann wählen, ob alle Aliens pro Angriffswelle auf einmal erscheinen und ob sie gleiche oder verschiedene Startpositionen haben sollen.

Das wohl schönste Feature des Games Designer ist die Möglichkeit, die Angreifer lebendig zu gestalten. Dazu werden von einem Alientyp vier ähnlich aussehende Sprites (siehe unten) definiert, die dann während der Bewegung der Reihe nach wechseln.

Die Sprites — das sind aus einer Menge von einzelnen Punkten bestehende Zeichen, die sich über den Bildschirm bewegen (zum Beispiel Raketen, Raumschiffe ...) — können über die Option 3) mit Hilfe eines Zeichengenerators definiert werden. Der augenblickliche Zustand ist dabei jederzeit zu sehen; die Zeichenfarbe kann frei gewählt werden. Es sind bis zu 32 verschiedene Sprites möglich, die aber alle ihre feste Funktion haben. So ist für je zwei Angriffswellen nur ein Alientyp möglich, der jedoch »belebt« werden kann. Die Sprites haben eine Größe von 12 x 12 (!) Pixels; ihre Bewegung ist fließend.

Mit Hilfe dieser Optionen ist es also möglich, ein eigenes Arkadespiel zu erstellen, ohne irgend etwas über eine Programmiersprache zu wissen. Jedoch sind gute Englischkenntnisse notwendig, da das Programm wie das 24 Seiten lange beiliegende Manual in Englisch verfaßt sind. Trotz einer Menge schöner Möglichkeiten hat der Games Designer jedoch auch viele Einschränkungen und Nachteile, die schon auffallen, wenn man nur die schon vorprogrammierten Beispiele anschaut. Es ist nicht möglich, außer der eigenen Kanone und den Aliens noch irgend etwas anderes darzustellen, wie zum Beispiel eine Landschaft für ein »echtes« Scramble, oder Mauern und andere Hindernisse, wie man sie von Space Invaders her kennt. Keines dieser beiden Spiele läßt sich mit dem Games Designer auch nur annähernd realisieren, weil man zum Beispiel keine Raketen aufstellen kann, die plötzlich starten, da jedes feindliche Schiff immer in Bewegung sein muß und es am rechten Rand wieder auftaucht, wenn es den Bildschirm links verläßt. Es ist auch eine eigentlich so simple Bewegung der Angreifer wie beim »echten« Space Invaders nur schwer zu verwirklichen, da gleichbleibende Aliens immer neben- beziehungsweise übereinander angeordnet sind und sowieso nicht mehr als acht von ihnen am Bildschirm Platz haben. Überdies kann die Geschwindigkeit innerhalb einer Angriffswelle nicht geändert werden. Außerdem sind eine Menge anderer Funktionen und Bedingungen fest vorgegeben, so daß doch alle hier entwickelten Spiele ziemlich gleich aussehen. Doch würden weitere Optionen die sowieso schon etwas komplizierte und vor allem ziemlich zeitraubende Entwicklung wohl noch weiter erschweren, so daß im Rahmen des Möglichen und Vernünftigen und für den Preis (14,95 Pfund = zirka 60 Mark) ein recht guter Kompromiß gefunden wurde.

Für den Programmieranfänger ist der Games Designer sicherlich recht interessant, doch auch für diesen, wie für den fortgeschrittenen Hobby-Programmierer zeigen sich schnell die Grenzen auf und der Reiz daran geht bald verloren. Nicht nur deshalb, weil die hier entwickelten Spiele zum alten Eisen gehören und die Spielmotivation — auch in Anbetracht der neuen faszinierenden Spiele, die laufend auf den Markt kommen — sehr schnell verlorengeht. (Th. Stögmüller)

Fortsetzung von Seite 100

```
2980 IF O=2 THEN 3010
2990 IF O=3 THEN 3100
3000 IF O=4 THEN 3220
3010 P$=LEFT$(P$,1):L$(M)=P$
3020 IF L$(M)="C" THEN E=0:GOTO 3270
3030 IF L$(M)="D" THEN E=1:GOTO 3270
3040 IF L$(M)="E" THEN E=2:GOTO 3270
3050 IF L$(M)="F" THEN E=3:GOTO 3270
3060 IF L$(M)="G" THEN E=4:GOTO 3270
3070 IF L$(M)="A" THEN E=5:GOTO 3270
3080 IF L$(M)="B" THEN E=6:GOTO 3270
3090 IF L$(M)="R" THEN E=-1:GOTO 3270
3100 P$=LEFT$(P$,2):L$(M)=P$
3110 IF L$(M)="+C" THEN E=7:GOTO 3270
3120 IF L$(M)="+D" THEN E=8:GOTO 3270
3130 IF L$(M)="+E" THEN E=9:GOTO 3270
3140 IF L$(M)="+F" THEN E=10:GOTO 3270
3150 IF L$(M)="+G" THEN E=11:GOTO 3270
3160 IF L$(M)="+A" THEN E=12:GOTO 3270
3170 IF L$(M)="#C" THEN E= 0:GOTO 3270
3180 IF L$(M)="#D" THEN E= 1:GOTO 3270
3190 IF L$(M)="#F" THEN E= 3:GOTO 3270
3200 IF L$(M)="#G" THEN E= 4:GOTO 3270
3210 IF L$(M)="#A" THEN E= 5:GOTO 3270
3220 P$=LEFT$(P$,3):L$(M)=P$
3230 IF L$(M)="+#C" THEN E= 7:GOTO 3270
3240 IF L$(M)="+#D" THEN E= 8:GOTO 3270
3250 IF L$(M)="+#F" THEN E=10:GOTO 3270
3260 IF L$(M)="+#G" THEN E=11:GOTO 3270
3270 SKIP -E
3280 CURSOR0,12:PRINT" FA Note Nr. ";M;"
 von ";G;" FA"
3290 IF L$(M)="R" THEN PCOLOR3:PRINT/PTA
B(F);"P":GOTO 3320
3300 PCOLOR2:IF H$="#"THEN PCOLOR3
3310 PRINT/PTAB(F);"O"
3320 E=E-1:SKIP+E
3330 IF I$="1"THEN I$="16":GOTO 3380
3340 IF I$="3"THEN I$="8":GOTO 3380
3350 IF I$="5"THEN I$="4":GOTO 3380
3360 IF I$="7"THEN I$="2":GOTO 3380
3370 IF I$="9"THEN I$="1"
3380 SKIP+2:PRINT/PTAB(F);I$:SKIP-3
3390 IF(M=38)+(M=76)+(M=114)+(M=152)*(M=
190)+(M=228)THEN PRINT/P:PRINT/P:PRINT/P
:GOTO 2930
3400 IF M=G THEN PRINT"@":GOTO 2450
3410 GOTO 2960
3420 PRINT/P:PRINT/P:F=0:FORJ=1TO5
3430 PCOLOR1:MODE5
3440 PRINT/P
3450 PRINT/P"----------------------------
----------------------------------------
-------------"
3460 NEXT
3470 RETURN
```

***Bild 1. Listing »Komponieren« (Schluß)***

# Nachhall

Grafik-Hardcopy: Bilder einfach ausdrucken für Colour Genie
Happy-Computer 2/84, Seite 126

Leider wurde das Listing verwechselt. Bei Bedarf schickt Ihnen Hr. Höhne (Tel. 04381/6697) ein korrektes Listing zu.

Korrektur zu »Vor Madagaskar«, Happy-Computer 2/84, Seite 96 bis 102:

☐ Am Ende der Zeile 35 muß angefügt werden ::T=12

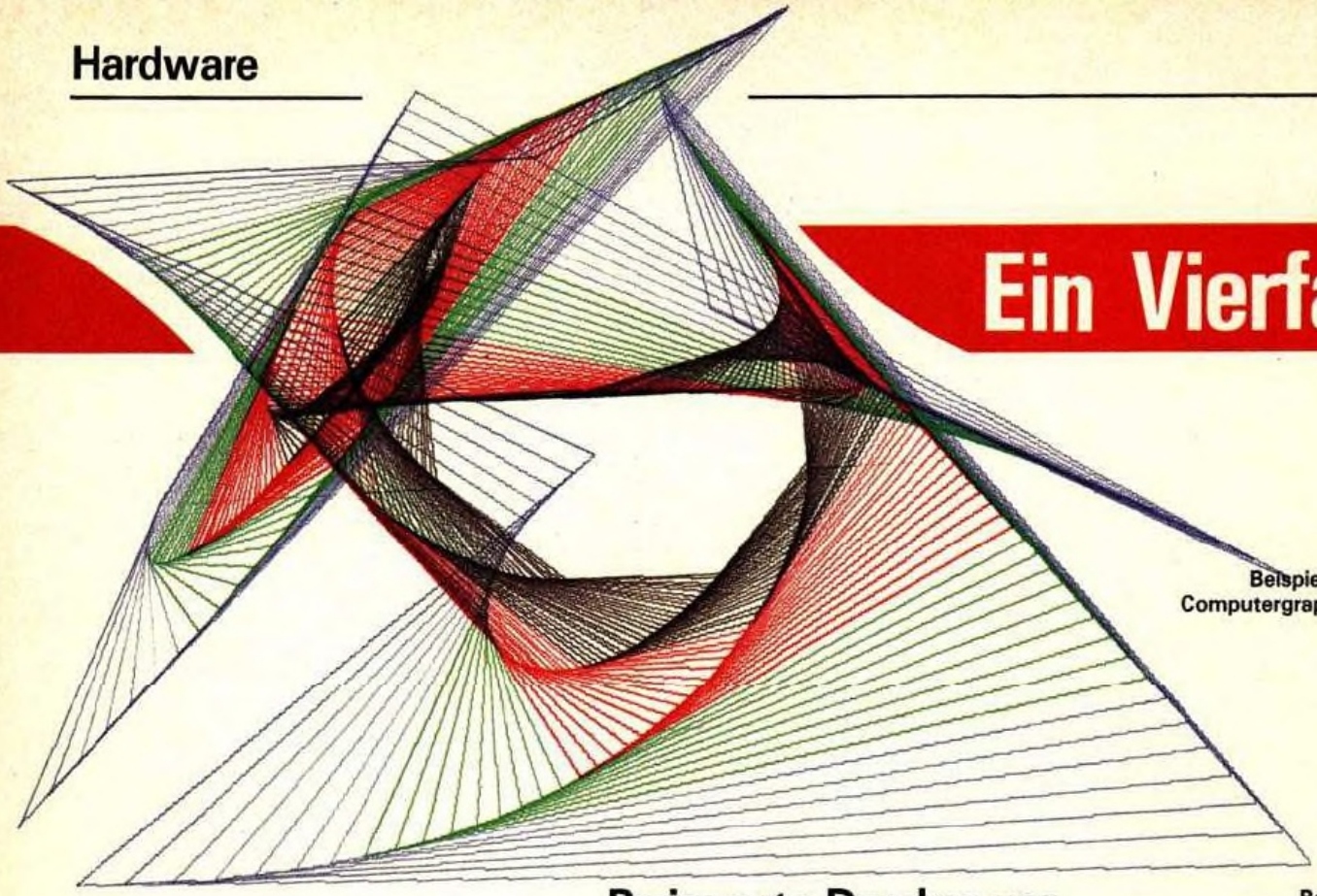
☐ In Zeile 5042 fehlt der Separator "::" nach dem ersten "'NEXT I"

☐ Der Rücksprungbefehl "SUBEND" muß aus Zeile 5657 herausgenommen und in eine gesonderte zeile geschrieben werden:

5657 IF TA >30 THEN CALL MUSIK :: RESTORE 4090 :: CALL TAFEL :: CALL HCHAR (18,132,32) :: CALL VCHAR (20+0,32,34) :: 0=0+1 :: TA=4
5660 SUBEND
660 CALL VISIER1(VI,TORP) :: IF K=32 AND T >0 THEN CALL TORPEDO (GR,VI,T,TORP)
5700 SUB VISIER1(VI,TORP)
5705 CALL KEY(0,K,S) :: IF S=0 OR TORP=1 THEN SUBEXIT

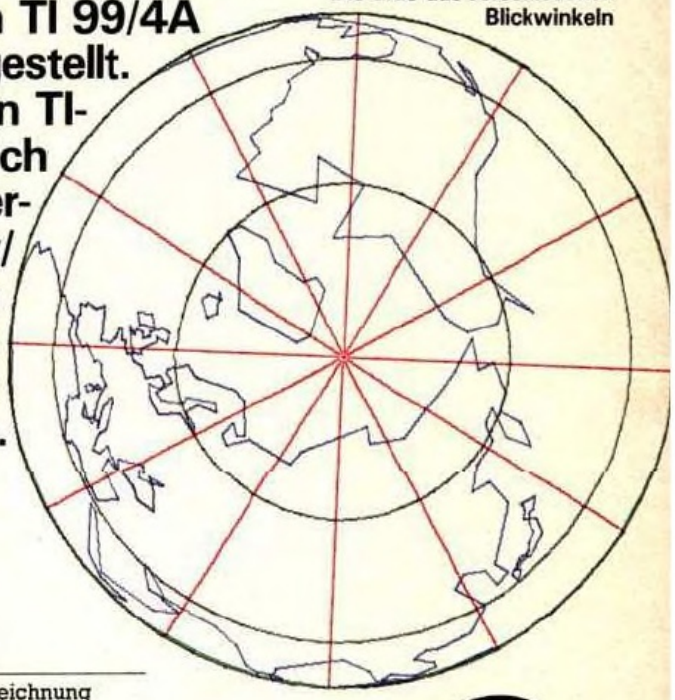
Dafür sind die Zeilen 5270 und 5280 ersatzlos zu streichen.

# Ein Vierfarbp



Beispiel 6:
Computergraphik

**Preiswerte Drucker von Texas Instruments für den TI 99/4A werden bekanntlich nicht mehr hergestellt. Da andere Drucker für die meisten TI-Besitzer wohl zu teuer sind, bietet sich als Alternative zum Beispiel der Vierfarbplotter »CGP-115« von Tandy/Radio Shack an. Im folgenden werden Anschluß und Betrieb beschrieben.**



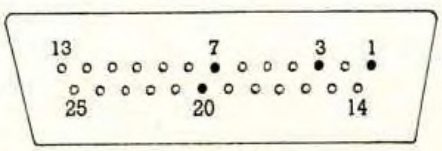
Beispiel 4:
Die Erde aus verschiedenen Blickwinkeln

Der Trommelplotter »TRS-80 CGP-115« erhöht die Verwendbarkeit Ihres TI 99/4A-Homecomputers durch zusätzliche Plotterkapazität und die Möglichkeit zum Auflisten von Programmen. Der Anschluß des CGP-115-Plotters ist leider nicht ohne weiteres möglich. Erforderlich sind das Extended Basic-Modul und die Peripherie-Erweiterungsbox mit der RS232C-Karte.

Die Leitungen und die Steckerbelegung für serielle Datenübertragung sind international genormt; dennoch ist das Standard-TRS-Kabel nicht verwendbar. Das Kabel muß über einen DB-25-Stecker für den Anschluß an die RS232C-Schnittstelle und einen 4poligen DIN-Stecker für den Anschluß am TRS-80 CGP-115-Plotter verfügen. Die Verbindung vom TI 99/4A-Homecomputersystem zum Plotter läuft über nur drei Leitungen plus Abschirmung. Die Stifte sind wie folgt belegt:

| Stift RS232C | Stift 4-DIN | Bezeichnung |
|---|---|---|
| 1 | | Schutzerde |
| 3 | 4 | Sendedaten |
| 7 | 3 | Betriebserde |
| 20 | 2 | Betriebsbereitschaft |
| | 1 | Nicht verwendet |

DB25 Stecker:




4-DIN Stecker:

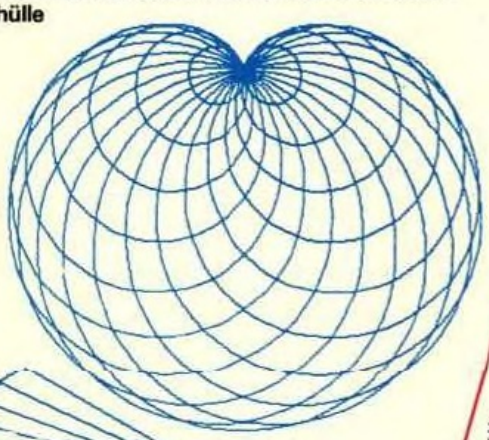
Folgende Einstellungen der DIP-Schalter sind vorzunehmen: Serieller Printer, 40 Zeichen, Carriage Return, 7 Bit ASCII. Nach dem Einsetzen der Farbstifte muß man Papiervorschubschalter und Ein/Aus-Schalter gleichzeitig drücken. Hiernach führt der CGP-115-Plotter einen Selbsttest aus, indem er Qua-

# lotter für den TI 99/4A

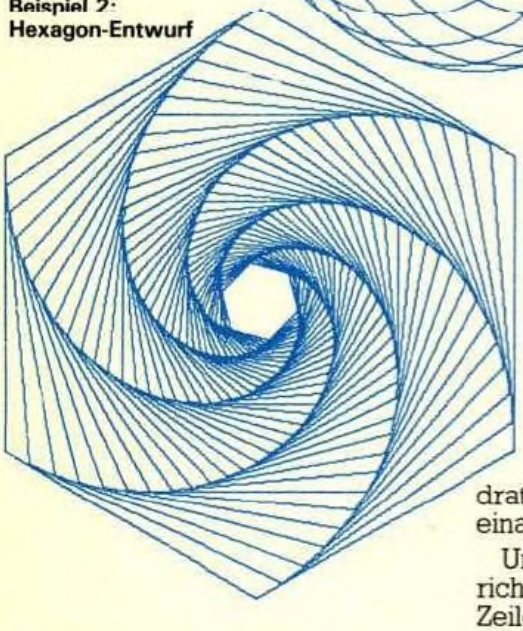
Beispiel 4:
Die Erde aus verschiedenen Blickwinkeln

```
Kant 1:                    Kant 2:
TI99/4A+CGP-115
Funktionen  000-009
Geburtstagprogramm
            010-017
Hexagon Entwurf            LEER
            020-024
Kardioid Entwurf
        .   030-036



4 TI99/4A PROGRAMM@

TI99/4A & CGP-115 PROGRAMMEN
FUNKT. GEB.TAG,HEXAGON,KARD@          PLT

     AUTHOR:M.Hooijberg
     Doeverensestraat 27
     NL-4265-JM  GENDEREN/HOLLAND
```

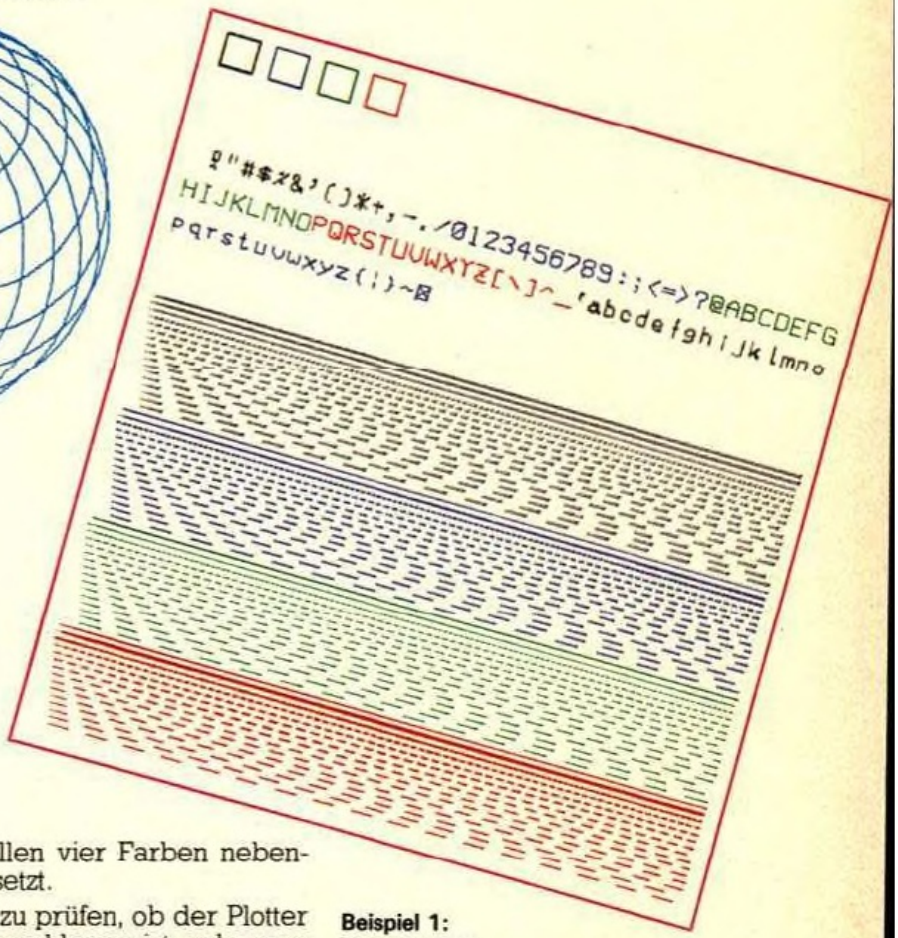
Beispiel 5: Kassettenhülle



Beispiel 3:
Kardioid-Entwurf



Beispiel 2:
Hexagon-Entwurf



Beispiel 1:
Verschiedene Linienarten und Farben

drate in allen vier Farben nebeneinander setzt.

Um nun zu prüfen, ob der Plotter richtig angeschlossen ist, gebe man Zeile 40 bis 160 des Programms auf der folgenden Seite ein und liste es mit

# Ein Vierfarbplotter für den

```
10 ! TI99/4A+RS232+CGP-115PLOTTER
20 ! ############################
30 ! LIST  "RS232.BA=600.PA=N.TW.LF":170
-910
40 OPEN #1:"RS232.BA=600.PA=N.TW.LF"
50 PRINT #1:CHR$(18)
60 FOR F=0 TO 3
70 FOR I=0 TO 15
80 PRINT #1:"I"
90 PRINT #1:"D0,0,480,0","L";I
100 PRINT I;
110 PRINT #1:"L";I
120 PRINT #1:"C";F
130 PRINT #1:"M0,0,0,-5"
140 NEXT I :: NEXT F
150 CLOSE #1
160 END
170 !
180 !##################
190 !### TEXT MODE ###
200 !##################
210 ! Zeichen ruecksprung
220 PRINT #1:CHR$(8);
230 ! Zeile vorwaerts
240 PRINT #1:CHR$(10);
250 ! Zeile ruecksprung
260 PRINT #1:CHR$(11);
270 ! Text   Mode
280 PRINT #1:CHR$(17)
290 ! Farbstift wechseln
300 PRINT #1:CHR$(29)
310 !
320 !##################
330 !## GRAFIK MODE ##
340 !##################
360 FOR VERZOEGERUNG=1 TO 600 :: NEXT VE
RZOEGERUNG
366 ! Grafik Mode
370 PRINT #1:CHR$(18)
380 ! Farbstift heben, Origin ist ganz
390 !  links + Plotter in Text Mode
400 PRINT #1:"A"
410 ! Wechseln Farben "C0",.."C3"
420 PRINT #1:"Cx"
430 !Stift senken und zeichnen (Absolute
)
440 PRINT #1:"D"
450 ! Zurueck nach Origin
460 PRINT #1:"H"
470 ! Origin waehlen
480 PRINT #1:"I"
490 !Stift senken und zeichnen (Relativ)
500 PRINT #1:"J"
510 ! Strichart waehlen "L0",.."L15"
520 PRINT #1:"Lx"
530 ! Stift heben und bewegen (Absolute)
540 PRINT #1:"M"
550 ! Text mit Grafik Mode
560 PRINT #1:"P"
570 ! Text Richtung waehlen "Q0",.."Q3"
580 PRINT #1:"Qx"
590 ! Stift heben und bewegen (Relativ)
600 PRINT #1:"R"
610 ! Zeichengroesse  "S0",.."S63"
620 PRINT #1:"Sx"
630 ! Achsen zeichnen:X0=Vert.,X1=Hor.
640 PRINT #1:"Xx"
650 !##################
660 !### BEISPIELE ###
670 !##################
680 PRINT #1:"C";I !(I=0 TO 3)
690 PRINT #1:"C3"
700 PRINT #1:"D240,0"
710 PRINT #1:"HD";X;",";Y
720 PRINT #1:"D";D(K+1);",";Y
730 PRINT #1:"J0,480,480,0,0,-480,-480,0
"
740 PRINT #1:"L15"
750 PRINT #1:"L";S !(S=0 TO 15)
760 PRINT #1:"M130,0,130,";(N-1)*15
770 PRINT #1:"M480,-999"
780 PRINT #1:"M";D(K+1);",";Y
790 PRINT #1:"M0,";J
800 A$="M"
810 PRINT #1:A$;X;",";Y :: A$="D"
820 PRINT #1:"IP   TI99/4A"
830 PRINT #1:"P";N$ !(N$="TEXT")
840 PRINT #1:"Q";R !(R=0 TO 3)
850 PRINT #1:"R80,-200"
860 PRINT #1:"S";Z !(Z=0 TO 63)
870 PRINT #1:"S10"
880 PRINT #1:"X0,10,10" ! (Vert)
890 PRINT #1:"HX1,10,10" ! (Hor)
900 CLOSE #1
910 END
```

**Dieses Programm erzeugt den Ausdruck in Beispiel 1 (Zeilen 40 bis 160) und erläutert den Befehlssatz des Plotters.**

# TI 99/4A

LIST
»RS232.BA=600.PA=N.TW.LF«
Nach dem Run-Befehl zeichnet der Plotter eine Anzahl vierfarbiger Linien (Beispiel 1).

## Umfangreicher Befehlssatz

Der Plotter verfügt über zwei Betriebsarten:
1. den Text-Mode, einzuschalten mit PRINT # 1:CHR$(17);
2. den Grafik-Mode, anzuwählen mit PRINT # 1:CHR$(18).

Der Befehlssatz ist umfangreich (siehe Programm 1, Zeilen 170 bis 640). Im Grafik-Modus hat der CGP-115-Plotter einen Zeichenbereich von 480 x 0,2 mm = 96 mm (horizontal) und 2 x 999 x 0,2 mm = 199,8 mm (vertikal). Falls beide Betriebsarten in einem Programm öfters auftauchen, muß man mindestens eine halbe Sekunde Verzögerung einschalten, siehe Programm 1, Zeile 360.

Der TRS-80 CGP-115-Plotter hat sehr viele Einsatzmöglichkeiten, wie die abgedruckten Beispiele zeigen. Durch seinen relativ günstigen Preis ist er für den Hobbycomputerbereich sehr zu empfehlen.

(Martin Hooijberg)

# Nachhall

Spaß am Komponieren für den MZ 700: (Happy-Computer 3/84, Seite 99). Das abgedruckte Programm ist leider nicht vollständig. Es fehlen folgende Programmzeilen:

```
2180 IF Q=1 THEN Y=Y+1
2190 X=X+1:CURSOR Y,X:PRINT"| "
2200 IF M$ ="+C" THEN M$ = "c"
2210 IF M$ ="+D" THEN M$ = "d"
2220 IF M$ ="+E" THEN M$ = "e"
2230 IF M$ ="+F" THEN M$ = "f"
2240 IF M$ ="+G" THEN M$ = "g"
2250 IF M$ ="+A" THEN M$ = "a"
2260 IF M$ ="B"  THEN M$ = "H"
2270 IF M$ ="+#C" THEN M$ ="#c"
2280 IF M$ ="+#D" THEN M$ ="#d"
2290 IF M$ ="+#F" THEN M$ ="#f"
2300 IF M$ ="+#G" THEN M$ ="#g"
2310 IF Q=1THEN Y=Y-1
2320 COLOR,,7,1:CURSOR Y,13:PRINT M$
2330 IF A=1 THEN C$="16":C=1
2340 IF A=2 THEN C$="8" :C=2
2350 IF A=3 THEN C$="4" :C=4
2360 IF A=4 THEN C$="2" :C=8
2370 IF A=5 THEN C$="1" :C=16
2380 CURSOR Y,15:PRINT C$
2390 IF Q=1 THEN Y=Y+1
2400 TT$="0":NEXT
2410 CURSOR 0,16:FOR J=1 TO 8
```


HAPPY COMPUTER

**An alle Home-Computer-Fans, die von Computerspielen und Home-Software begeistert sind.**

Wir bieten Ihnen die Chance, in dem faszinierenden und geradezu explodierenden Spiele- und Home-Software-Markt mit dabei zu sein. Für unser erfolgreiches Magazin »Happy-Computer« suchen wir zur Verstärkung unseres **Redaktionsteams** einen Fachmann- oder eine Fachfrau für die Rubrik Spiele/Home-Software. Als Insider in diesem Bereich sollten Sie in der Lage sein, Trends auf diesem explosiven Markt zu erkennen, neue interessante Produkte zu finden, zu testen und in geeigneter Weise darüber zu berichten! Um Ihre vielfältigen Aufgaben erfolgreich erledigen zu können, sind Besuche nationaler und internationaler Messen erforderlich, gute Kontakte zu Herstellern und Händlern sowie Besuche von Pressekonferenzen unerläßlich.

Interessiert? Wir bieten Ihnen die Möglichkeit, in einem jungen, unkonventionellen Team Ihre berufliche Zukunft zu verwirklichen. Rufen Sie uns doch einfach an; Herr Nunner, Tel. 089/4613-132, informiert Sie vorab gerne. Ihre schriftliche Bewerbung richten Sie bitte an:

**Markt&Technik** Verlag, Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar bei München

# Mäuse Mäuse Mäuse im Computerlabyrinth Mäuse Mäuse

**Jedes Leben hat seine Schattenseiten. Das der Mäuse endet oft in einem Labyrinth: bei Stadtmäusen in der Kanalisation, bei unseren Mäusen im Würfellabyrinth des TI 99/4A (mit Extended Basic und Joysticks). Wenn Sie ihnen aber mit Geschick und kluger Taktik beistehen, haben Sie ihre Freude und die Computermäuse ein längeres Leben.**

Das Spielprinzip gleicht in gewisser Beziehung einem uralten Kinderspiel, bei dem in einem kleinen Rahmen Holzplättchen mit Zahlen oder Buchstaben solange verschoben werden mußten, bis sich eine bestimmte Reihenfolge ergab. Die Schwierigkeit, aber auch der Reiz, bestanden darin, daß in der Matrix von 6 x 6 Plättchen lediglich eine Lücke von einem Plättchen bestand, in die jeweils eines der umgebenden Plättchen geschoben werden konnte.

Mit einem Computer können wir uns natürlich die »Holz«-ware sparen und ein wenig Bewegung mit ins Spiel bringen. Denn ein zeitlich begrenzter Zugzwang erhöht die Spannung ungemein. Und die Rosa-Panther-Serenade zu Beginn jeder Spielrunde schafft zusätzlich Atmosphäre.

Das Spielfeld besteht aus einer 4 x 4 Matrix, in dem jedes Feldchen einen Teil eines Labyrinths darstellt. Ein Feld ist allerdings leer. Immer eine Maus kriecht mit gleichbleibender Geschwindigkeit durch das Labyrinth. Die Felder mit den Labyrinth-Teilen sind jedoch ungeordnet, so daß die Maus schon nach kurzer Zeit vor einem »Mauer«-teil zum Stehen käme und damit verloren wäre. Ziel des Spiels ist es nun, die Labyrinth-Teile durch ständiges Verschieben so umzustellen, daß für die Maus ein fortlaufender Gang entsteht. Dazu müssen Sie mit dem Joystick solange eines der Teile, die um das jeweils leere Feld angeordnet sind, auswählen und auf das leere Feld schieben bis die Maus freie Bahn hat. Drücken Sie zum Beispiel den Joystick nach oben, rutscht das Labyrinth-Teil unterhalb des Leerfeldes auf dessen Platz, während dadurch das untere Feld leer wird.

Im ganzen Labyrinth sind acht Gegenstände (Kirschen, Zitronen, Herzen und so weiter) verstreut, die von der Maus beim Passieren gefressen werden — man hört sogar das Schlürfgeräusch. Wenn alle Gegenstände auf diese Weise verschwunden sind, ist eine Runde erfolgreich beendet und es beginnt eine neue, mit höherem Kriechtempo der Maus, einem zusätzlichen »Bonus-Nager« und neuen Gegenständen.

Das Ganze klingt sehr leicht. Aber die unaufhaltsam auf den Rand irgendeines falsch plazierten Labyrinth-Teils zustrebende Maus wird Sie nach kurzer Zeit ganz schön ins Schwitzen bringen. Nur gut, daß Sie zu Beginn eines jeden Spiels vier Mäuse zum »Aufarbeiten« bekommen. Im entscheidenden Moment fehlt Ihnen nämlich ganz sicher das passende Teil. Ein Trick sei aber verraten: Wenn eine Maus sich gerade in dem Labyrinth-Teil befindet, den Sie bewegen, wird die Maus mit übertragen. Sollten Sie dennoch einmal alle Mäuse aufgebraucht haben, ist das Spiel beendet.

Der Programmaufbau enthält einige ungewöhnliche Detaillösungen und ist nicht ganz einfach. Zu Beginn des Programms werden sämtliche Sonderzeichen defi-

```
100 !  ********************
110 !  *                  *
120 !  * WUERFELLABYRINTH *
130 !  *                  *
140 !  ********************
150 !
160 CALL CLEAR :: RANDOMIZE :: CALL MAGNIFY(3)
170 DIM L(4,4,1)
180 DISPLAY AT(10,1):"SPIELANLEITUNG(J/N) ?"
190 CALL KEY(0,KE,ST):: IF ST<>0 THEN 190
200 IF KE=74 OR KE=106 THEN GOSUB 1520
210 CALL CLEAR :: CALL COLOR(8,2,3,9,2,3,10,2,3,11,2,3,13,2,3)
220 C1$="000000000001F3F7F7F7F7F3F3F1F000000000608102040804E0F0F0F0E0C000" ! KIR
SCHE
230 C2$="00000000030F1F3FFFFF3F1F0F03000000000000C0F0F8FCFFFFFCF8F0C0000000" ! ZIT
RONE
240 C3$="00001C3E7F7F7F7F3F1F0F0703010000000003B7CFEFEFEFEFCF8F0E0C0800000" ! HEK
Z
250 C4$="0000010101010101030707070F01000000808080808080C0E0E0E0E0F0800000" ! GLO
CKE
260 C5$="00000000000000010204080A192410080000038444424588000000000000000000" ! SCH
LUESSEL
270 C6$="0000000000020518D962F4344880C000000000000000048AD169F4C2221300000" ! SPI
NNE
280 CALL CHAR(96,"FFFFFFFFFFFFFFFF",97,"FFFFFFFF",98,"00000000FFFFFFFF",99,"F0F0
F0F0F0F0F0")
290 CALL CHAR(100,"0F0F0F0F0F0F0F0F",101,"00000000F0F0F0F0",102,"0000000000F0F0F0
F",103,"F0F0F0F0")
300 CALL CHAR(104,"0F0F0F0F",105,"F0F0F0F0FFFFFFFF",106,"0F0F0F0FFFFFFFFF",107,"
FFFFFFFF0F0F0F0F")
310 CALL CHAR(108,"FFFFFFFF0F0F0F0",109,"0000000080C0E0F0",110,"0000000001030707
F",111,"F0E0C080")
320 CALL CHAR(112,"0F0707030",113,"F0F0F0F0080402010",114,"80402010F0F0F0F0",115,"
0102040F0F0F0F0")
330 CALL CHAR(116,"0F0F0F0F102040801",117,"7F7F7F7F7F7F7F7F",118,"FEFEFEFEFEFEFEFE
E",119,"00FFFFFFFFFFFFFF")
340 CALL CHAR(120,"FFFFFFFFFFFFFF",121,"001F3F7F7F7F7F7F",122,"00F0FCFEFEFEFEFE
,123,"7F7F7F7F7F3F1F")
350 CALL CHAR(124,"FEFEFEFEFEFCF8",94,"F0F0F0F0F0F0F0F0",95,"F0F0F0F0F0F0F0F0",1
36,"01070F0307072713C")
360 CALL CHAR(140,"30C8E4E0C0F0E080",88,"0000204084B67E7F"&RPT$("0",48),128,"000
00402216D7EFE"&RPT$("0",48))
370 FOR L1=5 TO 17 STEP 4 :: B=B+1 :: A=0 :: FOR L2=7 TO 19 STEP 4 :: A=A+1 :: L
(B,A,0)=L2 :: L(B,A,1)=L1 :: NEXT L2 :: NEXT L1
380 FOR L1=1 TO 9 :: FOR L2=1 TO 4 :: READ CH$ :: CC$(L1,L2)=CH$ :: NEXT L2 :: N
EXT L1
390 ! SPIELBEGINN....
400 R=1 :: MA=4 :: GPU=0 :: SP$=RPT$(" ",16)
410 GOSUB 1240 :: DISPLAY AT(1,23):"PUNKTE" :: DISPLAY AT(3,23):USING "#####":GP
U :: DISPLAY AT(22,23):"MAEUSE" :: CALL HCHAR(24,25,88,4)
420 ZA=8
430 IF R=1 THEN T=1 :: V=7 :: PU=100 :: CALL CHAR(132,C1$)
440 IF R=2 THEN T=2 :: V=11 :: PU=200 :: CALL CHAR(132,C2$)
450 IF R=3 THEN T=2 :: V=7 :: PU=500 :: CALL CHAR(132,C3$)
460 IF R=4 THEN T=3 :: V=11 :: PU=500 :: CALL CHAR(132,C4$)
470 IF R=5 THEN T=3 :: V=16 :: PU=500 :: CALL CHAR(132,C5$)
480 IF R=6 THEN T=3 :: V=15 :: PU=1000 :: CALL CHAR(132,C6$)
490 CALL SPRITE(#10,132,V,9,124,#11,132,V,81,17,#12,132,V,121,41,#13,132,V,169,1
24)
500 CALL SPRITE(#14,132,V,81,225,#15,132,V,121,201,#16,132,V,17,73,#17,132,V,17,
170)
510 FOR L1=5 TO 20 :: DISPLAY AT(L1,7)SIZE(16):SP$ :: NEXT L1
520 DISPLAY AT(10,9)SIZE(12):"ES FOLGT NUN"
530 DISPLAY AT(13,11)SIZE(12):"RUNDE ";R
540 RESTORE 1740 :: FOR L1=1 TO 17 :: READ TI,FR :: CALL SOUND(TI*100,FR*2,1,FR,
9,FR/2,15):: NEXT L1
550 FOR L1=1 TO 4 :: FOR L2=1 TO 4 :: Y=L(L1,L2,1):: X=L(L1,L2,0):: A=INT(RND*9)
+1 :: K(L1,L2)=A
560 FOR L3=0 TO 3 :: DISPLAY AT(Y+L3,X)SIZE(4):CC$(A,L3+1):: NEXT L3
570 NEXT L2 :: NEXT L1
```

**Listing zum Mäuse-Spiel »Würfellabyrinth«**

```
580 A=INT(RND*2)+1 :: B=INT(RND*2)+1 :: Y=L(B,A,1):: X=L(B,A,0):: GOSUB 1100 ::
K(B,A)=0 :: P=B :: Q=A
590 YM=-T :: XM=0
600 CALL SPRITE(#1,136,16,70,21,YM,XM)
610 CALL JOYST(1,J1,J2):: GOSUB 680
620 IF J1+J2=0 THEN 610
630 IF J1=4 AND Q>1 THEN C=Q-1 :: D=P :: GOTO 1030
640 IF J1=-4 AND Q<4 THEN C=Q+1 :: D=P :: GOTO 1030
650 IF J2=4 AND P<4 THEN C=Q :: D=P+1 :: GOTO 1030
660 IF J2=-4 AND P>1 THEN C=Q :: D=P-1 :: GOTO 1030
670 GOTO 610
680 CALL COINC(ALL,CO):: IF CO THEN 1170
690 IF XM=T THEN 730
700 IF XM=-T THEN 800
710 IF YM=T THEN 870
720 IF YM=-T THEN 950
730 ! ..ABFRAGE-RECHTS..
740 CALL POSITION(#1,YP,XP)
750 CALL GCHAR(INT(YP/8)+1,INT(XP/8)+2,S)
760 IF S=107 OR S=112 THEN CALL SPRITE(#1,140,16,YP+8,(INT(XP/8)+1)*8-3,T,0):: Y
M=T :: XM=0 :: RETURN
770 IF S=100 OR S=116 THEN CALL SPRITE(#1,136,16,YP-8,(INT(XP/8)+1)*8-3,-T,0):: Y
M=-T :: XM=0 :: RETURN
780 IF S=117 OR S=32 THEN 1120
790 RETURN
800 ! ..ABFRAGE-LINKS..
810 CALL POSITION(#1,YP,XP)
820 CALL GCHAR(INT(YP/8)+1,INT(XP/8)+1,S)
830 IF S=108 OR S=111 THEN CALL SPRITE(#1,140,16,YP+8,(INT(XP/8)+1)*8-3,T,0):: Y
M=T :: XM=0 :: RETURN
840 IF S=99 OR S=113 THEN CALL SPRITE(#1,136,16,YP-8,(INT(XP/8)+1)*8-3,T,0):: Y
M=-T :: XM=0 :: RETURN
850 IF S=118 OR S=32 THEN 1120
860 RETURN
870 ! ..ABFRAGE-UNTEN..
880 CALL POSITION(#1,YP,XP):: IF YP>180 THEN 940
890 CALL GCHAR(INT(YP/8)+2,INT(XP/8)+1,S)
900 IF S=105 OR S=109 THEN CALL SPRITE(#1,88,16,(INT(YP/8)+1)*8-3,XP+8,0,T):: YM
=0 :: XM=T :: RETURN
910 IF S=115 OR S=98 THEN CALL SPRITE(#1,128,16,(INT(YP/8)+1)*8-3,XP+8,0,T):: YM
M=0 :: XM=-T :: RETURN
920 IF S=94 THEN CALL LOCATE(#1,1,(INT(XP/8)+1)*8-3)
930 IF S=119 OR S=32 THEN 1120
940 RETURN
950 ! ..ABFRAGE-OBEN..
960 CALL POSITION(#1,YP,XP):: IF YP<10 OR YP>184 THEN 1020
970 CALL GCHAR(INT(YP/8)+1,INT(XP/8)+1,S)
980 IF S=108 OR S=111 THEN CALL SPRITE(#1,88,16,(INT(YP/8)+1)*8-3,XP+8,0,T):: YM
=0 :: XM=T :: RETURN
990 IF S=113 OR S=97 THEN CALL SPRITE(#1,128,16,(INT(YP/8)+1)*8-3,XP+8,0,T):: YM
M=0 :: XM=-T :: RETURN
1000 IF S=95 THEN CALL LOCATE(#1,184,(INT(XP/8)+1)*8-3)
1010 IF S=120 OR S=32 THEN 1120
1020 RETURN
1030 E=K(D,C):: Y=L(P,Q,1):: X=L(P,Q,0)
1040 FOR L1=0 TO 3 :: DISPLAY AT(Y+L1,X)SIZE(4):CC$(E,L1+1):: NEXT L1
1050 GOSUB 680 :: Y=L(D,C,1):: X=L(D,C,0):: GOSUB 1100
1060 CALL POSITION(#1,Y1,X1):: IF Y1>185 THEN 1090
1070 CALL GCHAR(INT(Y1/8)+1,INT(XP/8)+1,S1):: IF S1<>32 THEN 1090
1080 CALL LOCATE(#1,Y1-J2*8,X1+J1*8)
1090 K(D,C)=0 :: K(P,Q)=E :: P=D :: Q=C :: GOTO 610
1100 ! ..LOESCHEN..
1110 FOR L1=Y TO Y+3 :: DISPLAY AT(L1,X)SIZE(4):"    " :: NEXT L1 :: RETURN
1120 ! ..FEHLER..
1130 MA=MA-1 :: CALL MOTION(#1,0,0)
1140 FOR L1=880 TO 220 STEP -10 :: CALL SOUND(10,L1,0):: NEXT L1
1150 CALL HCHAR(24,25,32,8):: IF MA=0 THEN 1660 ELSE CALL HCHAR(24,25,88,MA)
1160 GOTO 590
1170 CALL MOTION(#1,0,0):: ZA=ZA-1 :: CALL SOUND(200,-5,0)
1180 FOR L1=10 TO 17 :: CALL COINC(#1,#L1,20,C1):: IF C1 THEN CALL DELSPRITE(#L1
)
```

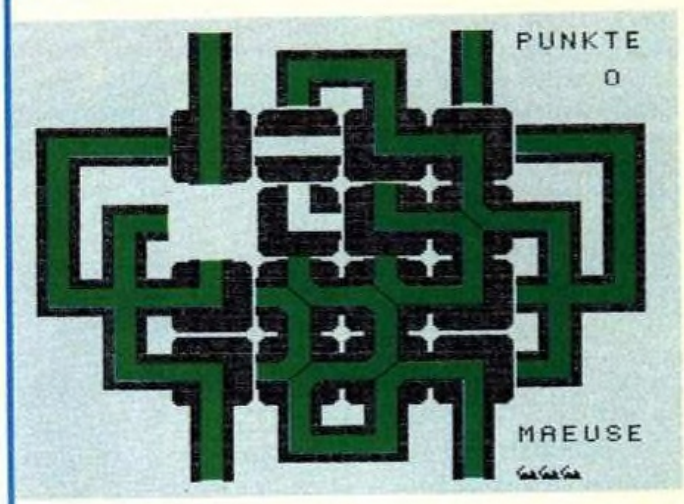
**Listing zum Mäuse-Spiel »Würfellabyrinth« (Fortsetzung)**

```
1190 NEXT L1
1200 GPU=GPU+PU :: DISPLAY AT(3,23):USING "#####":GPU
1210 IF ZA>0 THEN CALL MOTION(#1,YM,XM):: GOTO 690
1220 R=R+1 :: MA=MA+1 :: CALL HCHAR(24,25,88,MA):: IF R=7 THEN R=2
1230 GOTO 420
1240 ! ..ZEICHNE UMRANDUNG..
1250 DISPLAY AT(1,8):"cd              cd"
1260 DISPLAY AT(2,8):"cd  laaaak  cd"
1270 DISPLAY AT(3,8):"cd  cfbbed  cd"
1280 DISPLAY AT(4,8):"cd  cd  cd  cd"
1290 DISPLAY AT(6,1):"laaaaa"&SP$&"aaaaak"
1300 DISPLAY AT(7,1):"cfbbbb"&SP$&"bbbbed"
1310 DISPLAY AT(8,1):"cd    "&SP$&"    cd"
1320 DISPLAY AT(9,1):"cd    "&SP$&"    cd"
1330 DISPLAY AT(10,1):"cd laa"&SP$&"aak cd"
1340 DISPLAY AT(11,1):"cd cfb"&SP$&"bed cd"
1350 DISPLAY AT(12,1):"cd cd "&SP$&" cd cd"
1360 DISPLAY AT(13,1):"cd cd "&SP$&" cd cd"
1370 DISPLAY AT(14,1):"chagha"&SP$&"aghagd"
1380 DISPLAY AT(15,1):"ibbefb"&SP$&"befbbj"
1390 DISPLAY AT(16,4):"cd "&SP$&" cd"
1400 DISPLAY AT(17,4):"cd "&SP$&" cd"
1410 DISPLAY AT(18,4):"cha"&SP$&"agd"
1420 DISPLAY AT(19,4):"ibb"&SP$&"bbj"
1430 DISPLAY AT(21,8):"cd  cd  cd  cd"
1440 DISPLAY AT(22,8):"cd  chaagd  cd"
1450 DISPLAY AT(23,8):"cd  ibbbbj  cd"
1460 DISPLAY AT(24,8):"cd              cd"
1470 CALL HCHAR(1,10,95):: CALL HCHAR(1,22,95):: CALL HCHAR(24,10,94):: CALL HCH
AR(24,22,94)
1480 RETURN
1490 ! ..DATEN FUER QUADRATE
1500 DATA ycdz,ucdv,ucdv,{cd!,ywwz,aaaa,bbbb,{xx!,ycdz,agha,befb,{cd!,ycdz,ucha,
uibb,{xx!
1510 DATA ycdz,agdv,bbjv,{xx!,ywwz,aakv,bedv,{cd!,ywwz,ulaa,ucfb,{cd!,ycdz,aqpa,
bmrb,{cd!,ycdz,aota,bsnb,{cd!
1520 DISPLAY AT(1,5)ERASE ALL:"SPIELANLEITUNG"
1530 DISPLAY AT(4,1):"DAS SPIELFELD BESTEHT AUS   EINER REIHE VON KANAELEN,DIEZU
M GROSSTEIL ZUFAELLIG          BESTIMMT WERDEN."
1540 DISPLAY AT(9,1):"DER INNERE TEIL DES FELDES  IST IN 16 QUADRATE UNTER-   TE
ILT,DIE EINZELN BEWEGT         WERDEN KOENNEN."
1550 DISPLAY AT(14,1):"DIESE QUADRATE MUESSEN NUN  MIT HILFE DES JOYSTICKS SO  G
ESETZT WERDEN,DASS DIE MAUSDIE SICH IM LABYRINT BEWEGT"
1560 DISPLAY AT(18,1):"NIEMALS ZUM STEHEN KOMMT."
1570 DISPLAY AT(24,1):"DRUECKEN SIE EINE TASTE!" :: CALL KEY(0,KE,ST):: IF ST<=0
 THEN 1570
1580 CALL CLEAR
1590 DISPLAY AT(4,1):"DAS SPIEL BESTEHT AUS 6      RUNDEN."
1600 DISPLAY AT(7,1):"ZIEL JEDER RUNDE IST ES,ALLESYMBOLE DIE IM SPIELFELD    VE
RTEILT SIND ABZUFAHREN."
1610 DISPLAY AT(11,1):"NACH JEDER BESTANDENEN RUNDEERHAELT MAN EINE MAUS HINZU."
1620 DISPLAY AT(14,1):"DIE 'ALPHA-LOCK'TASTE MUSS  GELOEST WERDEN."
1630 DISPLAY AT(17,4):"VIEL GLUECK !!!"
1640 DISPLAY AT(24,1):"DRUECKEN SIE EINE TASTE!" :: CALL KEY(0,KE,ST):: IF ST<=0
 THEN 1640
1650 CALL CLEAR :: RETURN
1660 CALL CLEAR :: CALL DELSPRITE(ALL)
1670 DISPLAY AT(10,1):"SIE HABEN ALLE MAEUSE      VERBRAUCHT"
1680 DISPLAY AT(13,1):"WENN SIE NOCHMAL SPIELEN   WOLLEN DANN DRUECKEN SIE J  A
NSONSTEN N"
1690 CALL KEY(0,KE,ST):: IF ST<=0 THEN 1690
1700 IF KE=74 OR KE=106 THEN 400
1710 IF KE=78 OR KE=110 THEN STOP
1720 GOTO 1690
1730 END
1740 ! ..MUSIK-DATEN..
1750 DATA 7,391,1,440,7,466,1,369,3,391,1,440,3,466,1,622,3,587,1,391,3,466,1,58
7
1760 DATA 12,554,1,466,1,391,1,349,16,391
```

**Listing zum Mäuse-Spiel »Würfellabyrinth« (Schluß)**

```
L1,L2,L3...SCHLEIFENVARIABLEN
ZA.........RESTLICHE SYMBOLE
MA.........RESTLICHE MAEUSE
R..........MOMENTANE RUNDE
T..........GESCHWINDIGKEIT DER MAUS
V..........FARBE DER SYMBOLE
PU.........PUNKTE PRO SYMBOL
GPU........GESAMTPUNKTZAHL
J1,J1......JOYSTICKAUSLENKUNG
YM,XM......BEWEGUNGSRICHTUNG DER MAUS
SP$........LEERSTRING ZUM ZEICHNEN DER UMRANDUNG
C1$-C6$....CHARDATEN DER SYMBOLE
L(4,4,1)...POSITIONEN DER WUERFEL
CC$(9,4)...AUSSEHEN DER WUERFEL
K(4,4).....WELCHER WUERFEL AN WELCHER POSITION

SPRITES

#1.........MAUS
#10-#17....SYMBOLE

CHARS

88,128,136,140...MAUS IN 4 RICHTUNGEN
94-124...........SPIELFELD
132..............SYMBOL
```

**Variablenliste und Verzeichnis der Zeichencodes zu »Würfellabyrinth« Sprite Nummern**




**In der Hardcopy erkennt man die Labyrinth-Teile besonders gut**

```
210.........Bildschirm löschen,Farben setzen
220-360.....Chardefinition
370.........Einlesen der Würfelpositionen
380.........Einlesen der Würfelbilder
400.........Setzen der Anfangsbedingungen
430-480.....Rundenabhängige Anfangsbedingungen
490-500.....Setzen der Symbolsprites
510-540.....Schreiben des Rundenbeginns,Anfangsmusik
550-580.....Auswahl und Zeichnen von 15 Würfeln
600.........Maus startet
610-670.....Joystickabfrage und Verzweigungen
680.........Abfrage : Frucht erreicht?
690-1020....Abfrage nach Richtungsänderung
1030-1110...Verschieben eines Würfels
1130-1160...Fehlerauswertung
1170-1230...Frucht fressen,nach 8.Frucht neue Runde
1250-1480...Umrandung zeichnen
1500-1510...Daten über Würfelaussehen
1520-1650...Spielanleitung
1660-1730...Neues Spiel?
1750-1760...Daten für Anfangsmusik
```

**So ist das Programm »Würfel-labyrinth« aufgebaut**

niert. Auf eine Definition der Zeichen mit Hilfe von DATA-Statements wurde bewußt verzichtet, da der Programmieraufwand wegen der Möglichkeit, mehrere Characters in einem Befehl definieren zu können, gegenüber der Verwendung von DATA nicht wesentlich höher, jedoch die Ausführungsgeschwindigkeit und Übersichtlichkeit bei direkter Definition der Sonderzeichen wesentlich besser ist. Die Characters der Symbolsprites (Kirschen, Zitronen und so weiter) wurden in Stringvariablen abgespeichert, da sonst die Zahl der noch zur Verfügung stehenden Zeichen nicht ausgereicht hätte (ein Sprite besteht aus vier Characters. Außerdem wird pro Runde ja auch nur jeweils ein Symboltyp benötigt, der ohne Zeitprobleme gesondert definiert werden kann. Eine Besonderheit stellen die Characters 94 und 95 dar, die nur benötigt werden, um festzustellen, ob die Maus den Bildschirm nach oben oder unten verläßt.

Nach der Characterdefinition werden alle möglichen Positionen, die ein Labyrinth-Teil innerhalb des Spielfeldes einnehmen kann in das Feld L(4,4,1) geladen. In einem zweiten Feld CH$(9,4) werden dann alle neun Teiletypen abgespeichert. Im Feld K(4,4) steht schließlich, welche Form sich an welcher Stelle im Spielfeld befindet. Letzteres ist als Koordinatensystem zu sehen mit der Größe 4 x 4 Feldern. Wird nun zum Beispiel der Würfel an der Stelle 1,1 (linke obere Ecke) benötigt, so hat er folgende Daten:

Bildschirmposition
Y=L(1,1,1),X=L(1,1,0)
Würfelform E=K(1,1)

Verwendete Characters sind:
CH$(E,1),CH$(E,2),CH$(E,3),CH$(E,4)

Diese Art der Zuordnung ermöglicht ein übersichtliches und schnelleres Verarbeiten der Informationen.

Sowohl die Spielfeldumrandung als auch die einzelnen Würfel werden übrigens nur mit »DISPLAY AT«-Befehlen gezeichnet. Dies ist nicht nur ein geringerer Aufwand gegenüber der Verwendung von »CALL HCHAR«, es steigert auch die Ausführungsgeschwindigkeit um ein Vielfaches.

### Je höher die Spielrunde, desto schwieriger

Ein besonderes Problem stellte die Abfrage zur Richtungsänderung der Maus dar. Hier zeigten sich die Vor- und Nachteile eines Sprites. Denn die Programmierunabhängigkeit der Sprites ermöglicht zwar, daß schnelle Bewegungen einfach erzeugt werden können, jedoch erhöht sich mit steigender Spritegeschwindigkeit und mit steigender Anzahl der Befehle, die zwischen zwei Positionsabfragen liegen, die Gefahr, daß der Computer »zu spät« kommt und Fehler auftreten. Hier mußte also ein Kompromiß zwischen Geschwindigkeit und Spielkomfort gefunden werden. Die Geschwindigkeit der Mäuse wurde deshalb auf maximal drei gesetzt und die Abfrage möglichst kurz gehalten, um einen fehlerfreien Programmablauf zu garantieren. Wie Sie selbst sehen werden, ist diese Geschwindigkeit durchaus ausreichend, um in den höheren Runden das Spiel dennoch äußerst schwierig werden zu lassen.

Die Abfrage nach der Richtungsänderung der Maus funktioniert folgendermaßen: Zuerst wird die Position des Mausprites festgestellt. Dann wird an dieser Position eine »GCHAR«-Abfrage durchgeführt, um den weiteren Verlauf der Strecke festzustellen. Abhängig davon wird nun verzweigt und der Maus eventuell eine neue Richtung und ein neues Aussehen gegeben. Dies alles geschieht in einem von vier ähnlichen Unterprogrammen.

(Ch. Bernhard/W. Czerny)

# 2000 Mäuse für die Labyrinth Mäuse

Christian Bernhard, am 23.9.60 in München geboren, besucht die Fachhochschule München

**Eine originelle Spielidee und eine tadellose programmtechnische Umsetzung mit ungewöhnlichen Detaillösungen waren uns 2000 Mäuse für die beiden Autoren wert.**

Wolfgang Czerny, am 27.7.61 gleichfalls in München geboren, studiert ebenfalls an der Fachhochschule München

Auch dieses Mal sind die beiden hervorstechenden Eigenschaften des ausgezeichneten Programms die ausgefallene Programmidee und die hervorragende Umsetzung im Programm, das damit besser ist als die meisten Fertigprogramme und die Möglichkeiten des TI 99/4A voll ausreizt. Dazu kommen einige raffinierte und ungewöhnliche Programmtechniken.

Das Programm »Würfellabyrinth« gehört sowohl in die Kategorie der Denkspiele wie auch in die der Action-Spiele. Wer das Spiel gewinnen will, braucht nämlich taktisches Geschick ebenso wie ein gutes Reaktionsvermögen. Die beiden 24 und 23 Jahre alten Programmautoren Christian Bernhard und Wolfgang Czerny aus München formulieren ihre Ausgangsüberlegungen zu dem Programm so: »Es war unser Ziel, ein Programm zu schreiben, das möglichst alle positiven Eigenschaften eines guten Spielprogrammes in sich vereinigt. Also hohe Geschwindigkeit, gute grafische Darstellung, Übersichtlichkeit, relativ geringer Programmieraufwand (Speicherbedarf: zirka 11600 Byte) und musikalische Untermalung. Vor allem aber legten wir Wert darauf, das Programm möglichst interessant zu gestalten, es also zu einem Spiel zu machen, das man immer wieder gerne hervorholt. Dies ist uns, wie wir meinen, durch volles Ausnutzen der Möglichkeiten, die der TI 99/4A in Extended Basic bietet, gelungen.«

Die zwei Studenten der Fachhochschule besuchten 1979 gemeinsam eine Klasse der Fachoberschule in München und stellten schon damals fest, daß sie beide Interesse an Computern hatten. Von da an stand für sie fest, dieses Hobby später einmal zum Beruf machen zu wollen. Nach anfänglichen Versuchen auf den Schulcomputern PET 2001 und Wang entschlossen sie sich zum gemeinsamen Kauf des TI 99/4A. Nach und nach legten sich die beiden das komplette System (Disketten-Laufwerk, Drucker, Assembler, Pascal, Extended Basic und so weiter) zu. Stundenlanges, oft bis spät in die Nacht reichendes Programmieren führte bald zu einer Palette von unterschiedlichsten Programmen, wobei sie »irrgartenverwandte« Spiele schon immer besonders faszinierten. Deshalb entstand auch das vorliegende Programm. Derzeit arbeiten sie intensiv daran, ihre Programmierkenntnisse in Pascal und Assembler zu verbessern.

Wer mit den beiden als Gleichgesinnter in Kontakt treten will, kann sich an folgende Adresse wenden: Czerny Wolfgang, Aschheimer Str. 2, 8000 München 80

Unsere heutige Wahl soll Sie, lieber Leser, auch dazu ermutigen, bei Ihren eigenen Programmen von den ausgetretenen Pfaden abzugehen und eigene Ideen zu entwickeln. Daß es wieder ein Spiel war, welches Listing des Monats wurde, liegt vielleicht daran, daß die Zahl der eingesandten Spielprogramme erheblich höher ist als die Zahl der »ernsthaften Programme. Wir hoffen aber, bald ein Programm aus einer anderen Programmkategorie prämiieren zu können. (lg)

# Sternen-abenteuer

**»Rescue Ship« ist ein Weltraumabenteuer mit guter Grafik für den TI 99/4A unter Extended Basic. Es besteht aus drei Spielstufen. Schnelle Reaktion ist dafür nötig — und ein Joystick.**

TI 99/4A

Die einzelnen Stufen werden außer der ersten als Anwenderunterprogramme aufgerufen und definiert, von denen aus wieder Unterprogramme aufgerufen werden. Zeile 40 mit dem Befehl »ON WARNING STOP« dient der Sicherung gegen Ausfälle oder Störungen, die in der 16-KByte-Ausstattung aufgrund des Speicherbedarfs (vor »RUN ENTER« 10658 Byte) auftreten können. Dem gleichen Zweck dienen die RUN-Statements in den Zeilen 60 und 490. Besitzer der Speichererweiterung können diese Teile dementsprechend abändern. Die Bedeutung der Variablen geht aus dem Zusammenhang im jeweiligen Unterprogramm hervor.

Nach Eingabe des RUN-Befehls fragt der Computer nach dem ordnungsgemäßen Anschluß der Fernbedienung. Gespielt wird mit Joystick 1; durch Drücken des Feuerknopfs beginnt der Computer mit dem Hauptprogramm.

## Spieleziel: Keinen durchkommen lassen

Das eigene, weiße Raumschiff erscheint am linken Rand, das gelbe, feindliche Raumschiff nähert sich von rechts. Erreicht es unbeschädigt den linken Bildschirmrand, hat man verloren — es sei denn, die Punktezahl ist höher als 50, dann erhält der Spieler drei Raumschiffe. Er ist in diesem Fall erst besiegt, wenn dreimal das feindliche Raumschiff das Spielfeld passiert hat. Pro Abschuß eines Gegners erhält man 2 Punkte.

In der zweiten Stufe ändert sich außer der Farbe der Gegner (rot) auch deren Zahl (2) und Bewegung: Statt stur geradeaus zu fliegen, nehmen sie einen Zickzack-Kurs. Für zwei Abschüsse gibt es 400 Punkte; drei Raumschiffe erhält man ab 2900 Punkten. Innerhalb der zwei ersten Stufen sind die Bewegungsmöglichkeiten des eigenen Raumschiffs auf ein Rauf- und Runterfahren beschränkt; Seitwärtsbewegungen sind durch Ändern des MOTION-Befehls möglich. Beachten Sie aber, daß die Variablenwerte der Joysticks ±4 betragen. Die höchste Zahl, mit der man X und Y multiplizieren kann, ist außerdem 31.

## Ufos, Meteoriten und schwarze Löcher

Das Bewegungsmuster der dritten Stufe ist komplizierter: Sie können nicht einfach stehenbleiben, sondern haben eine Grundbewegung (vergleiche Zeile 1190). Die dritte Stufe ist nicht nur die schwierigste, sondern auch die entscheidende. Hier werden die

```
10 !*************************************************************
   *****   *****RESCUE SHIP*****************************************
   *****      ****************************************************
   *****           ************************************************
   *****                  *****************************************
20 !*****         *************************************************
   *****      ****************************************************
   *****   *******************************************************
   ***************************************************************
30 !*COPYRIGHT BY KNUTH GÖTZ*UHLANDSTRASSE 9*7990 FRIEDRICHSHAFEN/BO
DENSEE*26.10.1983*************************************************
40 ON WARNING STOP
50 CALL CLEAR :: CALL SD :: CALL V1 :: DEF H=INT(RND*145)+10 :: N=RND5
60 Z=Z+1 :: IF Z=51 THEN RUN 70 ELSE 80
70 CALL S2 :: STOP
80 HR=H :: CALL LOCATE(#4,HR,232) :: CALL LOCATE(#3,HR,245) :: CALL
COLOR(#4,12)
90 CALL MOTION(#4,0,-28) :: CALL L1(4,Z,TT) :: CALL POSITION#4,X,Y)
:: RANDOMIZE X*Y :: N=RND :: GOTO 60
100 STOP
110 SUB VL1
120 FOR I=30 TO 0 STEP -1 :: CALL SOUND(-20,I*50+110,I/2,-7,I/2) ::
NEXT I :: CALL COL16
130 DISPLAY AT(10,1)BEEP:"sie haben ihren auftrag nicht erfuellt!die
hauptbasis ist vernichtet!"
140 fori=31 TO 1000 :: NEXT I :: DISPLAY AT(10,1)ERASE ALL:"deshalb
wird ihr raumschiff zerstoert!" :: CALL CG(1,500) :: CALL T :: STOP
150 SUBEND
160 SUB T
170 DEF F=RND*13+3 :: DEF X=RND*196+1 :: DEF Y=RND*252-126 :: FOR I=
1 TO 28 :: RANDOMIZE I :: CALL SPRITE(#I,138,F,X,X,Y,Y) :: NEXT I
180 FOR I=1 TO 4000 :: NEXT I :: CALL CLEAR
190 SUBEND
200 SUB L1(I,Z,TT)
210 CALL JOYST(1,X,Y) :: CALL MOTION(#1,-Y*6,0) :: CALL KEY(1,
W,S) :: IF S=0 THEN 260
220 CALL SOUND(-4000,500,22,1600,22-1,22) :: CALL POSITION(#1,
X,Y) :: CALL LOCATE(#2,X,Y)
230 FOR Q=1 TO 3 :: CALL COINC(#2,#I,18,C) :: IF C=-1 THEN CAL
L C(I) :: CALL SOUND(-4000,110,22,146,22,155,22,-7,22) :: GOTO
250
240 NEXT Q :: CALL LOCATE(#2,250,10) :: GOTO 260
250 CALL LOCATE(#2,240,10) :: DISPLAY AT(1,1):USING"##":Z*2 ::
SUBEXIT
260 CALL COINC(#I,#3,12,C) :: IF (C=-1 AND Z<26)OR TT=3 THEN C
AL L VL1 ELSE IF C=1 THEN TT=TT+1
270 GOTO 210
280 SUBEND
290 SUB SPD
300 CALL DELSPRITE(ALL)
310 CALL SPRITE(#1,142,16,82,10,0,0) : CALL SPRITE(#2,143,16,2
40,10,0,126) :: CALL SPRITE(#3,134,2,100,248,0,0)
320 FOR I=4 TO 10 :: CALL SPRITE(#I,138,1,1,1,Q-28) :: NEXT I
330 SUBEND
340 SUB CP
350 CALL CHAR(142,"6010FEFEFC060C") :: CALL CHAR(138,"1836FFFFF
F3C42") :: CALL CHAR(134,"10103854FE38")
360 CALL CHAR(135,"FFBABABABABABAB") :: CALL COLOR(13,7,2) ::
CALL SCREEN(2)
370 CALL HCHAR(24,1,135,32) :: CALL HCHAR(23,3,135,24) :: CALL
HCHAR(22,6,135,19) :: CALL CHAR(21,15,135,6)
380 CALL VCHAR(19,17,135,20) :: CALL CHAR(136,"F7F7F7F7F7F7F7F
7") :: CALL COLOR(14,5,16) :: CALL VCHAR(1,31,136,72)
390 SUBEND
400 SUB S2
410 P=100 :: CALL MOTION(#4,0,0)
420 D1=RND*12-6 :: D2=RND+12-6 :: CALL SPD :: DEF H=RND*90+50
:: CALL SPRITE( 4,13,4,2,100,248,D2,0) :: CALL MOTION(#3,D1,0)
430 D1=RND*12-6 :: D2=RND*12-6
440 CALL POSITION(#6,X,Y) :: RANDOMIZE X*Y :: H1=H :: H2=H
450 CALL MOTION(#7,D1,-10,#6,D2,-10,#3,D1,0,#4,D2,0) :: CALL L
OCATE(#7,H1,230,#5,H1,250) :: CALL LOCATE(#6,H2,230,#4,H2,250)
460 CALL COLOR(#7,7,#,6,7) :: P=P+400 :: CALL L2(7,6,D1,D2,P,T T)
470 I=I+1 :: IF I 15 THEN 430
480 DISPLAY AT(13,1):"sie haben die basis verteidigt.starten s
ie nun, um den gegner endgueltig zu vernichten und die kolonie
zu retten."
490 FOR I=1 TO 28 :: CALL CG(F,1) :: NEXT I :: CALL CLEAR :: R UN 500
500 CALL S3
510 SUBEND
520 SUB VL2
530 CALL COL16
540 DISPLAY AT(12,1)ERASE ALL:"sie haben sich tapfer gewehrt,
aber der gegner war staerker. sie werden zerstoert!"
550 CALL CG(1,5) :: CALL SOUNDS(0) :: CALL T :: STOP
560 SUBEND
570 SUB L3(E2,Z2,D1,D2,P,TT)
580 E1,E2,BX,BY=0 :: SEIT1,SEIT2=-10
590 CALL JOYST(1,X,Y) :: CALL MOTION(#1,-Y*10,0) :: CALL KEY(1
,W,S) :: IF S<>0 THEN CALL SOUND(-4000,16000,22,4000,22,500,22
,-2,22) ELSE 660
600 CALL POSITION(#1,X  ,Y) :: CALL LOCATE (#2,X,Y) :: FOR I =
1 TO3 :: CALL COINC(#2,#7,25,BX) :: IF BX=-1 THEN CALL C(7) EL
SE 620
610 D1,SEIT1=0
620 CALL JOYST(1,X,Y) :: CALL MOTION(#1,-Y*10,0)
```

Listing zu »Rescue Ship«

```
630 CALL COINC(#2,#6,25,BY) :: IF BY=-1 THEN CALL C(6) :: D2,S
HIT2=0
640 X1=X1+BX :: X2=X2+BY :: IF (NOT X1=0)AND (NOT X2=0) THEN C
ALL LOCATE(#2,240,1) :: DISPLAY AT(1,1):P :: SUBEXIT
650 NEXT I :: CALL LOCATE(#2,240,10)
660 CALL POSITION(#7,X,Y) :: IF Y>=160 OR Y<=55 THEN D1=-D1 :: CALL
MOTION(#7,D1,SHIT1,#5,D1,0) :: CALL RELOC(7,3,D1)
670 CALL JOYST(1,X,Y) :: CALL MOTION(#1,-Y*10,0)
680 CALL POSITION(#6,X,X) :: IF Y>=160 OR Y<=55 THEN D2=-D2 :: CALL
MOTION(#6,D2,SHIT2,#4,D2,0):: CALL RELOC(6,4,D2)
690 CALL COINC(#3,#7,18,X) :: CALL COINC(#4,#6,18,Y) :: IF X=-1 OR Y
=-1 THEN TT=TT+1 :: IF P<2900 OR TT=4 THEN CALL VL2
700 GOTO 590
710 SUBEND
720 SUB C(Z)
730 CALL COLOR(#Z,16) :: CALL COLOR(#Z,1) :: CALL SOUND(-4000,
110,22,300,22,17000,22,-8,22)
740 CALL POSITION(#Z,Y,X) :: CALL LOCATE(#Z,240,X) :: CALL MOT
ION( Z,0,0)
750 SUBEND
760 SUB S3
770 CALL POSITION(#INT(RND*28)+1,X,Y) :: CALL CLEAR :: CALL DE
LSPITE(ALL)
780 A=X*Y :: RANDOMIZE A :: CALL SCREEN(2) :: CALL COL16
790 B="sie bewegen sich mit hoher geschwindigkeit durch das all
auf die gegnerische basis hin."
800 A="ACHTUNG!meteoritenschwaerme koennen blitzschnell auftauc
hen, black holes sie aufsaugen."
810 DISPLAY AT(8,1)ERASE ALL: SEG$(A,1,255)"P.S.: black holes koennen ni
cht vernichtet werden. weichen sie aus." :: CALL SD
820 DISPLAY AT(24,1): "PRESS FB1 TO ENTER" :: CALL COLOR(12,2,2
)
830 S=0 :: CALL KEY(1,W,S) :: IF S=0 THEN 830 ELSE CALL CLEAR
840 CALL CHAR(137,"0") :: CALL COLOR(13,7,6) :: CALL HCHAR(24,1
,137,64)
850 X=0 :: CALL SCREEN(7) :: CALL CLEAR :: CALL COLOR(1,2,2) ::
CALL CP1(VA) :: CALL COLOR(12,13,2)
860 FOR I=20000 TO 0 STEP-500 :: CALL HCHAR(24,1,137,32) :: DIS
PLAY AT(24,1) SIZE(10): USING "#####miles": I :: H=RND*2000+1
870 IF 500-X<=0 THEN CALL V3 ELSE RANDOMIZE H
880 H1=INT(RND*5)+1 :: IF H1=5 THEN 890 ELSE ON H1 GOSUB 900,
910,920,930
890 NEXT I :: CALL S4(P-X)
900 DISPLAY AT(24,15)SIZE(2): "MS" :: CALL M(H,X,P) :: RETURN
910 CALL B(H,P) :: RETURN
920 DISPLAY AT(24,15) SIZE(2): "US" :: CALL U(H,X,P) :: RETURN
930 DISPLAY AT(24,15)SIZE(2):"GM" :: CALL G(H,X,P) :: RETURN
940 SUBEND
950 SUB SD
960 CALL CHAR(143,"1010103838FE") :: CALL CHAR(130,"1010") :: C
ALL CHAR(135,"8A787F70A0202020") :: CALL CHAR(136,"78F0A2E8A2F8
FCF8")
970 CALL SPRITE(#1,143,16,230,122) :: CALL SPRITE(#2,130,1,240,
100,-126,0)
980 DEF XV=INT(RND*170)-85
990 FOR I=3 TO 28 :: RANDOMIZE I :: CALL SPRITE(#I,135,14,82,122,XV,
XV) :: NEXT I
1000 SUBEND
1010 SUB B(X,P)
1020 P=P+6000 :: DEF S=INT(RND*6)-3 :: RANDOMIZE X*SQR(X) :: I=INT(R
ND*25)+3
1030 CALL COLOR(#I,2) :: CALL SOUND(-4000,110,22,110,22,198,19,-8,22
) :: RANDOMIZE I :: CALL MOTION(#I,S,S)
1040 FOR Q=1 TO 15 :: CALL JOYST(1,X,Y) :: CALL MOTION(#1,-Y*7,X*7)
:: CALL COINC(#1,#I,50,C) :: IF C=-1 THEN CALL VL3
1050 CALL DISTANCE(#1,#I,C2) :: DISPLAY AT(24,19)SIZE(5):USING"###.#
":SQR(C2) :: NEXT Q :: CALL SOUND(-4250,440,25,220,22,880,22,-1,22)
1060 CALL COLOR(#1,14) :: RANDOMIZE I :: CALL MOTION(#I,XV,XV)
1070 SUBEND
1080 SUB U(X,H,P)
1090 P=P+2000 :: RANDOMIZE LOG(X) :: I=INT(RND*25)+3
1100 CALL COLOR(#I,7) :: CALL PATTERN(#I,138)
1110 CALL TH(I,H,P) :: CALL COLOR(#I,14) :: CALL PATTERN(#I,135)
1120 SUBEND
1130 SUB M(X,H,P)
1140 P=P+1000 :: RANDOMIZE X*LOG(X*PI) :: I=INT(RND*25)+3
1150 CALL COLOR(#I,12) :: CALL PATTERN(#I,136)
1160 CALL TH(I,H,P) :: CALL COLOR(#I,14) :: CALL PATTERN(#I,135)
1170 SUBEND
1180 SUB TH(Z,H,P)
1190 CALL JOYST(1,X,Y) :: CALL MOTION(#1,-Y*7-25,X*7) :: CALL KEY(1,
W,S) :: IF S=0 THEN 1230
1200 H=H+1 :: CALL SOUND(-4000,16000,22,8000,22,-1,22) :: CALL POSITIO
N(#1,X,Y) :: CALL LOCATE(#2,X,Y) :: CALL COLOR(#2,16)
1210 FOR I=1 TO 3 :: CALL COINC(#2,#Z,20,C) :: IF C=-1 THEN H=H+2 ::
CALL COLOR(#2,1) :: CALL SOUND(-1000,110,22,440,22,-4,22) :: SUBEXIT
1220 NEXT I
1230 CALL COLOR(#2,1) :: FOR I=1 TO 2 :: CALL COINC(#1,#2,10,C) :: IF
C=-1 THEN P=P-2000 :: CALL SOUND(-4000,110,20,110,20,15000,20,-8,15)
1240 NEXT I :: IF P<=0 THEN CALL VL2 ELSE 1190
1250 SUBEND
1260 SUB G(H,X,P)
1270 CALL U(H,X,P) :: CALL M(H,X,P) :: CALL B(H,P)
1280 SUBEND
1290 SUB COL16
1300 FOR I=1 TO 14 :: CALL COLOR(I,16,1) :: NEXT I
1310 SUBEND
1320 SUB V3
1330 CALL CLEAR :: CALL COL16 :: DISPLAY AT(13,1):"sie haben nicht meh
r genug energie,um sich zu verteidigten.BEFEHL ZUR SELBSTZERSTOERUNG"
1340 CALL SOUNDS(-1) :: CALL CG(16,1) :: STOP
1350 SUBEND
1360 SUB CG(Z,X)
1370 FOR I=1 TO X :: NEXT I :: FOR I=16 TO 1 STEP -1 :: CALL COLOR(#Z
,I) :: NEXT I
1380 SUBEND
1390 SUB V1
1400 DATA RESCUE SHIP,! SPIELER,created&designed by,knuth goetz 1983&,
FFFFFFFFFFFFFFFF
1410 CALL COL16 :: CALL SCREEN(2) :: RESTORE 1400 :: READ A$,B$,C$,C2$
,D$ :: FOR I=1 TO 20 :: DISPLAY AT(I,I):A$ :: NEXT I :: CALL CHAR(140,
D$)
1420 CALL VCHAR(6,8,140,11) :: CALL VCHAR(7,13,140,4) :: CALL HCHAR(6,
8,140,6) :: CALL HCHAR(11,8,140,6)
1430 FOR I=1 TO 5 :: CALL HCHAR(11+I,8*I,140) :: NEXT I
1440 FOR I=6 TO 17 STEP 5 :: CALL HCHAR(I,18,140,5) :: NEXT I
1450 CALL VCHAR(7,18,140,4) :: CALL VCHAR(12,22,140,4)
1460 DISPLAY AT(18,7):B$ :: DISPLAY AT(22,2):C$,C2$
1470 FOR I=1 TO 16 :: FOR D=2 TO 14 :: CALL COLOR(D,I,2) :: NEXT D ::
NEXT I
1480 DATA "UEBERPRUEFEN SIE,OB DIE FERNBEDIENUNG RICHTIG ANGESCHLOSSEN
UND DAS ALPHA-LOCK AUSGESCHALTEN IST ."
1490 RESTORE 1480 :: READ A$ :: DISPLAY AT(13,1)ERASE ALL:A$ :: CALL S
PD
1500 CALL KEY(1,W,S) :: IF S=0 THEN 1500 ELSE CALL CP1(V2) :: CALL CP1
(V2) :: CALL CP
1510 SUBEND
1520 SUB S4(PUNKTE)
1530 DISPLAY AT(13,1)ERASE ALL:"SIE HABEN DEN GEGNER GESCHLAGEN,WOIMMER
SIE IHN TRAFEN.SIE STEUERN AUF DIE BEFREITE KOLONIE ZU."
1540 DISPLAY AT(17,1):"DORT WERDEN SIE IN DIE EHRENRUNDE AUFGENOMMEN--
WENN SIE GENUEGEND PUNKTE HABEN."
1550 CALL SD :: CALL CP0 :: CALL CHAR(143,"80808FBFAFB7080") :: CALL SPR
ITE( 1,143,16,75,10,0,40)
1560 CALL COLOR(14,14,1) :: FOR I=1 TO 4 :: FOR I2=1 TO 5 :: CALL MOTIO
N(#I2+I*5,0,0) :: CALL SPRITE(#I2+I*5,143,7,47+I2*12,130,0,19) :: NEXT I2
1570 NEXT I :: CALL SOUND(-4000,440,22,330,22,16000,22,-2,22)
1580 IF PUNKTE =10000 THEN A$=" SIEGER 3.KLASSE" ELSE IF PUNKTE =15000
THEN A$=" SIEGER 2.KLASSE" ELSE A$=" SPITZENKLASSE"
1590 DISPLAY AT(13,1):"sie sind"&A$&"."
1600 OPEN #1:"CS1",INPUT,INTERNAL,FIXED 64
1610 CALL CLEAR :: FOR I=1 TO 5 :: INPUT #1:B$(I),B(I) :: DISPLAY AT(I
*3,1):B$(I),B(I) :: NEXT I
1620 CLOSE #1
1630 CALL HCHAR(15,1,32,300) :: ACCEPT AT(16,1)SIZE(14):B$(6)
1640 FOR I=1 TO 5
1650 IF B(I) PUNKTE THEN GOSUB 1670
1660 IF B(I)=PUNKTE THEN 1700 ELSE 1690
1670 FOR J=1 TO 5 :: IF B(I) B(J) THEN RETURN
1680 NEXT J :: B(I)=PUNKTE :: B$(I)=B$(6)&A$ :: GOTO 1700
1690 NEXT I :: IF B(I)  PUNKTE THEN 1720
1700 OPEN #1:"CS1",OUTPUT,INTERNAL,FIXED 64
1710 CALL CLEAR :: FOR I=1 TO 5 :: PRINT #1:B$(I),B(I) :: DISPLAY AT(I
*3,1):B$(I),B(I) :: NEXT I :: CLOSE #1:: GOTO 1730
1720 DISPLAY ERASE ALL:"sie sind leider nicht unter den ersten 5.":: F
OR I=1 TO 1000 :: NEXT I
1730 DISPLAY AT(13,1)ERASE ALL:"DANKE,SIE WAREN EIN GUTER GEGNER.AUF W
IEDERSPIELEN."
1740 FOR I=1 TO 2000 NEXT I :: CALL CLEAR :: CALL DELSPRITE(ALL) :: CA
LL T :: FOR I=28 TO 1 STEP -1 :: CALL CG(I,100) :: NEXT I
1750 CALL CLEAR :: SUBEND
1760 SUB CP0
1770 CALL CLEAR :: FOR I=1 TO 30  :: CALL VCHAR(24-INT(I/4)-1,I,140,
INT(I/4)+1) :: CALL VCHAR(1,I,140,INT(I/4)+1) :: NEXT I
1780 SUBEND
1790 SUB SOUNDS(A)
1800 L=-(A*201)+1 :: FOR I=2 TO 16 STEP 2 :: CALL SOUND(L,I*1000,-A,110,
-A,60+I*50,-A*5,-(I/2),-A) :: NEXT I
1810 SUBEND
1820 SUB RELOC(A,B,C)
1830 CALL POSITION(#A,Y,X) :: IF C=0 THEN Y=210 :: GOTO 1850 ELSE IF
Y =160 THEN Y=159
1840 IF Y >55 THEN Y=56
1850 CALL LOCATE(#A,Y,X,#B,Y,245)
1860 SUBEND
1870 SUB VL3
1880 FOR I=2 TO 16 :: CALL SCREEN(I) :: NEXT I :: CALL T :: CALL CP1(0)
1890 FOR I=2 TO 16 :: CALL SCREEN(I) :: CALL COLOR(1,1,18-I) :: CALL
COLOR(12,13,1) :: NEXT I :: GOTO 1890
1900 SUBEND
1910 SUB CP1(X)
1920 CALL CHAR(123,"FF00") :: CALL CHAR(124,"8080808080808080FF808080
8080808080")
1930 CALL HCHAR(1,1,123,800) :: FOR L=24 TO 2 STEP -2 :: X=X+1 :: CA
LL VCHAR(X,X,124,L) :: CALL HCHAR(X,X,125) :: NEXT L
1940 X=0 :: FOR L=24 TO 2 STEP -2 :: X=X+1 :: CALL VCHAR(X,33-X,124,
L) :: CALL HCHAR(12+X,19+X,32) :: NEXT L
1950 CALL COLOR(12,13,1) :: CALL HCHAR(1,1,32,32) :: CALL VCHAR(1,32
,32,24)
1960 SUBEND
```

**Listing zu »Rescue Ship« (Schluß)**

| | |
|---|---|
| S2<br>S3 | Spielstufen |
| S4 | Siegerehrung |
| V1<br>VL1<br>VL2<br>VL3<br>V3 | Spielbeginn<br>angesteuert bei<br>Niederlage |
| L1<br>L2 | Steuerprogramme |

Alle anderen Unterprogramme dienen zur Definition von Sprites, Grafik und Sound.

**Liste der Unterprogramme**

Punkte gesammelt, die später in die Bewertung eingehen. Statt Raumschiffen begegnen Sie einem Sterngewirr, aus dem sich Ufos (2000 Punkte), Meteoriten (1000) und schwarze Löcher (6000) rekrutieren, während Sie 20000 Meilen durch das All fliegen.

## Nach 500 Schuß ist Schluß

Angezeigt werden der Weg, den Sie noch zurücklegen müssen, Ufos und Meteoriten. Schwarze Löcher (Black Holes) kündigen sich akustisch nur mit Lärm und optisch nur indirekt mit der Anzeige der Entfernung an; sinkt diese unter 61, werden Sie von den Black Holes aufgesogen und haben verloren. Schwarze Löcher können natürlich nicht vernichtet werden. Halten Sie also besser Distanz — auch zu Ufos und Meteoriten. Bei Kollisionen mit diesen kostet Sie das ein neues Raumschiff 2000 Punkte; schießen Sie auch nicht blind um sich, Sie haben nur 500 Schuß. Sind diese verbraucht, erhalten Sie den Befehl zur Selbstzerstörung.

Sollten Sie sich wider Erwarten in die vierte und letzte Stufe retten, können Sie sich in die Runde der besten Fünf eintragen. Für diese Stufe muß eine Datei gemäß dem OPEN-Befehl in den Zeilen 1600 und folgenden vorhanden sein. Gegebenenfalls muß sie zuvor durch ein Hilfsprogramm erstellt werden (vergleiche die Zeilen 1700 und folgende). (Knuth Götz)

# Wildwasser wasserfahrt

**Ich wollte schon immer einmal ein Reaktions- und Geschicklichkeitsspiel schreiben, das wie kommerzielle Spiele mit dem Joystick gesteuert wird. Es sollte einen sportlichen Charakter haben und etwas abenteuerlich wirken. Als ich dann im Fernsehen einen Bericht über Wildwasserkanufahrer sah, hatte ich einen Kanu-Wettbewerb in ein Spielprogramm umgesetzt.**

Ziel des Spieles ist es, mit einem Kanu möglichst weit durch einen schnellen, kurvigen Fluß zu fahren, ohne auf das Ufer oder im Wasser liegende Felsen aufzulaufen. Dabei dürfen die beiden Paddel zwar die Steine, nicht aber das Ufer (den Wald) berühren.

Das Programm läuft auf dem Commodore 64 und ist rund vier KByte lang. Die wichtigsten Aufgaben werden durch Maschinenspracheroutinen erledigt, die im Basicprogramm als Datazeilen enthalten sind. Die erste Routine initialisiert die neu definierten Zeichen (Bäume, Felsen), die Farben und den Ton (Explosion). Die zweite Routine löscht den Bildschirm, druckt die Statuszeile (Anzeige Kanus und Meterzahl) und schaltet das Kanu (Sprite) ein. Das dritte Maschinenprogramm stellt das eigentliche Spiel dar. Es setzt den Fluß jeweils eine Reihe weiter nach unten, übernimmt die Joysticksteuerung des Kanus, erhöht die Punktezahl (Meter) und prüft auf Kollisionen mit Steinen oder dem Ufer. Ist das Kanu aufgelaufen, wechselt es seine Farbe, erzeugt eine Art Aufprallgeräusch, erniedrigt die Anzahl der Kanus und wartet auf die Feuertaste des Joysticks. Dann wird der Bildschirm teilweise gelöscht und das Spiel geht weiter.

Da das Programm in Basic geschrieben wurde, kann das Spiel fast immer mit

```
C U R S O R S T E U E R Z E I C H E N   IM LISTING:

ZEICHEN   BEDEUTUNG                    ERSCHEINUNG AUF DEM BILDSCHIRM (REVERS)

B         DUNKEL BLAU (CTRL-7)          PFEIL NACH LINKS
❤         BILDSCHIRM LOESCHEN           HERZCHEN
↓         CURSOR NACH UNTEN             BUCHSTABE 'Q'
S         CURSOR HOME                   BUCHSTABE 'S'

/         SONDERZEICHEN (NEU DEF.)      PFUNDZEICHEN
BEISPIELZEILE:
```

**Erklärung der im Listing verwendeten Steuerzeichen**

**Variablentabelle**

| | |
|---|---|
| I | Schleifenvariable |
| P | einzupokender Wert |
| S | Prüfsumme |
| W$ | Eingabestring |
| F$(1—3) | Namen der Flüsse |
| F | Nummer des Flusses (—1) |
| M$ | Anzahl der gefahrenen Meter |

**Speicherzellen**

247 maximale Anfangsspalte der Flüsse
248 Verzögerungswert für die Geschwindigkeit
249 Wert für Anzahl der Felsen
250 Konstante für Steuerung (8)
704—766 Daten für Kanu-Sprite
832—879 Daten für den Flußverlauf
(kopiert nach 680—695)
49152—49679 Maschinenspracheprogramm darin
49344 Anzahl der Kanus (+112)
49551 Joystickportkonstante
49578 Joystickportkonstante
49654 Joystickportkonstante

**Liste der verwendeten Variablen und Bedeutung bestimmter Speicherzellen**

Als nächstes ist der Computer am Zug

# »Awari« — ein Brettspiel für zwei

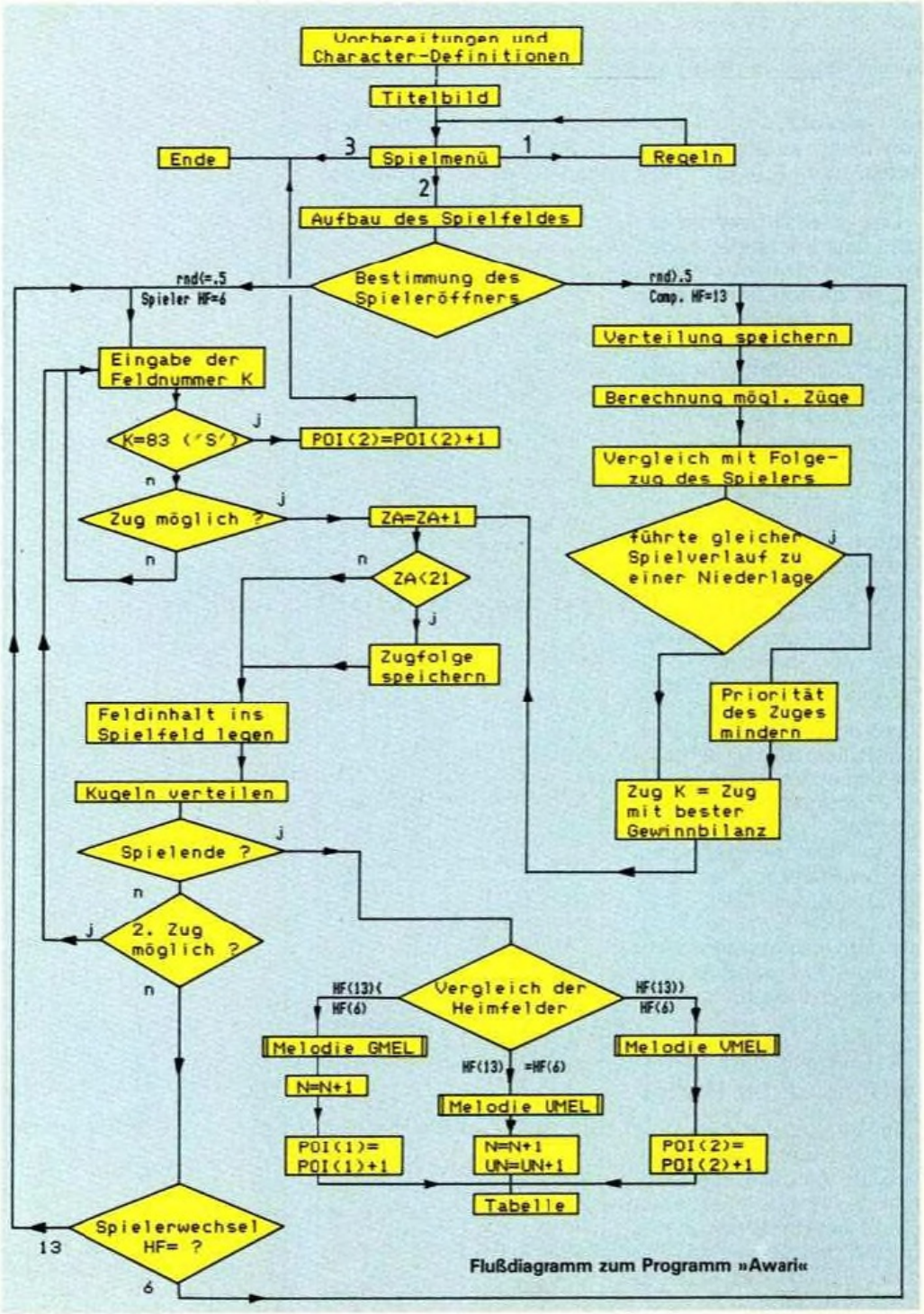
**Bei »Awari« handelt es sich um ein Brettspiel für zwei Personen. Sollte Ihnen ein passender Partner fehlen, aber ein TI 99/4A mit Extended-Basic-Modul verfügbar sein, dann versuchen Sie doch einfach Ihr Glück gegen den Computer.**

TI 99/4A

Das Spielfeld besteht aus 14 Feldern, die so angeordnet sind, daß beide Kontrahenten über jeweils sechs Felder verfügen, die gegenüberliegend aufgereiht sind. Diese werden von links nach rechts fortlaufend durchnumeriert. So liegen sich beispielsweise Feld 1 des Spielers A und Feld 6 des Spielers B gegenüber. Zusätzlich besitzt jeder Teilnehmer ein eigenes »Heimfeld«, das sich rechts neben Feld 6 befindet.

Zu Beginn der Partie erhält jeder Spieler 36 Spielsteine, die gleichmäßig auf die dazugehörigen sechs Spielfelder aufgeteilt werden. Ziel des Spieles ist, durch geschickt ausgeführte Züge möglichst viele Spielsteine im eigenen Heimfeld zu sammeln, wobei die Spieler abwechselnd an der Reihe sind.

Ein Zug besteht darin, sämtliche Steine eines eigenen Feldes zu entnehmen und diese dann einzeln, entgegen dem Uhrzeigersinn, in die folgenden Felder zu setzen. Landet der zuletzt abgelegte Stein eines Zuges in einem leeren Feld, während das gegenüberliegende besetzt ist, so erobert der Spieler die Steine beider Felder, das heißt sie werden in sein Heimfeld gelegt. Sollte andererseits der zu-



Flußdiagramm zum Programm »Awari«

| | | |
|---|---|---|
| F( ) | — | Inhalt der Spielfelder 1 - 14 |
| C( ) | — | dient dem Computer als Zwischenspeicher für F( ), um möglichst Züge zu berechnen |
| POI(1) | — | Anzahl gewonnener Spiele des Spielers |
| POI(2) | — | Anzahl gewonnener Spiele des Computers |
| UN | — | Anzahl unentschiedener Spiele |
| SC(1) | — | Gesamtzahl gewonnener Steine des Spielers |
| SC(2) | — | Gesamtzahl gewonnener Steine des Computers |
| L$(N) | — | speichert die ersten 20 Züge der laufenden Partie N |
| N | — | Zählindex der nicht vom Computer gewonnenen Partien |
| ANZ | — | Anzahl der Steine des Zugfeldes |
| ZA | — | Zählindex der aufgeführten Züge pro Runde |
| ENDE | — | 0 = Spiel fortsetzen, 1 = Spielende |
| ZUG | — | 0 = 1. Zug, 1 = 2. Zug |
| HF | — | Heimfeldnummer des am Zug befindlichen Spielers (6 = Spieler, 13 = Computer) |
| BI | — | Entscheidungshilfe für Computerzug |

**Variablenliste zu dem Spiel »Awari«**

letzt gesetzte Stein im eigenen Heimfeld landen, dann erhält der Spieler einen zweiten Zug.

Das Spiel ist beendet, sobald eine der beiden Seiten keine Steine mehr enthält. Sieger ist derjenige, dessen Heimfeld die Mehrzahl an Spielsteinen aufweist.

Das Computerprogramm, das aus Gründen der Übersichtlichkeit in mehrere Blöcke aufgeteilt wurde, beginnt mit einem vorbereitenden Teil (bis Zeile 300), der die »DIMENSION« Anweisungen sowie die Zeichen-Definitionen beinhaltet. Der Präsentation des Titelbildes (Zeile 310 bis 410) folgt nach Betätigen einer beliebigen Taste ein Spielmenü (Zeile 420 bis 490). Zur Auswahl stehen a) die Abfrage der Spielregeln — abgelegt in den Zeilen 1900 bis 2230, b) Start einer neuen Runde und c) Verlassen des Programms.

Das Spiel selbst beginnt mit der Erstellung der Spielfeldgrafik (Zeile 500 bis 750), der sich die Wahl des eröffnenden Spielers anschließt — diese wird dem Zufallsgenerator überlassen.

## S-Taste führt zum Abbruch der Partie

Im Verlauf der Partie hat der Spielteilnehmer lediglich die Aufgabe, die Nummer des Feldes einzugeben (Taste 1 bis 6), dessen Steine gesetzt werden sollen. Um den Ablauf des Spieles überschaubarer zu gestalten, werden anschließend sämtliche Steine des betreffenden Feldes, in Form von kleinen Karos, in die Mitte

```
100 ! ***  A W A R I  ***
110 !
120 !     Version 1.3
130 !
140 !    copyrights by
150 ! Hans-Juergen Adler '83
160 !
170 !   EAGLE-computing
180 !
190 !
200 DIM F(13),C(13),L$(30)
210 ! Vorbereitungen *****
220 CALL CHAR(135,"3C4299A1A199423C
    ")! copyright
230 CALL CHAR(128,"FFFFFFFFFFFFFFFF
    ",129,"1F3F7FFFFFFFFFFF")
240 CALL CHAR(120,"",130,"F8FCFEFFF
    FFFFFF",140,"10101010101010I0"
    )
250 CALL CHAR(131,"FFFFFFFFFF7F3F1F
    ",132,"FFFFFFFFFFFEFCF8")
260 CALL CHAR(64,"0000030003030")!
     Doppelpunkt ******
270 CALL CHAR(113,"00183C7E7E3C18",
    143,"F88080F08384FC03")! Kugel
    + Ec
280 CALL CHAR(37,"CC33CC33CC33CC33"
    ,142,"0000FFFF")! Bildschirm +
    Unterstr.
290 CALL CHAR(91,"0044003844447C44"
    ,92,"0044007C4444447C",93,"0044
    004444444438")
300 RANDOMIZE :: ON BREAK NEXT :: C
    ALL MAGNIFY(2)
310 !
320 ! Titelbild ++++++++++
330 CALL CLEAR :: CALL COL(11,11,2)
340 FOR I=1 TO 24 :: CALL HCHAR(I,1
    ,37,32):: NEXT I
350 DISPLAY AT(4,9)SIZE(-11):" A W
    A R I "
360 DISPLAY AT(23,1)SIZE(-23):" ";C
    HR$(135);" EAGLE-computing '83
    "
370 CALL COL(2,11,2)
380 CALL KII(13,127,K):: CALL CLEAR
390 ! 1-6 umdefinieren ***
400 FOR I=49 TO 54 :: CALL CHARPAT(
    I,B$):: CALL CHAR(I+48,B$):: NE
    XT I
410 FOR I=49 TO 54 :: CALL CHARPAT(
    I,B$):: CALL CHAR(I+55,B$):: NE
    XT I
420 !
430 ! Spielmenue +++++++++
440 CALL CLEAR :: CALL COL(2,10,5)
450 CALL P(3,5,"BITTE WAHLEN SIE:")
    :: CALL P(4,5,RPT$(CHR$(142),17
    ))
460 CALL P(12,6,"1 SPIELREGELN"):::
     CALL P(14,6,"2 NEUES SPIEL")
470 CALL P(16,6,"3 ENDE")
480 CALL KII(49,51,K):: IF K=83 THE
    N 1870
490 ON K-48 GOTO 1900,500,1870
500 !
510 ! Spielfeld ++++++++++
520 CALL CLEAR :: CALL SCREEN(1)
530 CALL COLOR(1,7,12,2,7,12):: CAL
    L COLOR(3,2,7,4,2,7)
540 FOR I=5 TO 8 :: CALL COLOR(I,2,
    11):: NEXT I
550 CALL COLOR(9,7,12,10,2,11,11,16
    ,11,12,1,5,13,7,12)
560 A$=RPT$("X",15):: CALL P(2,8,A$
    ):: CALL P(4,8,A$)
570 CALL P(3,8,"X A W A R I X")
580 A$=" "&CHR$(129)&CHR$(128)&CHR$
    (130):: CALL P(7,3,RPT$(A$,6))
590 CALL P(17,3,RPT$(A$,6)):: A$="
    "&RPT$(CHR$(128),3)
600 CALL P(8,3,RPT$(A$,6)):: CALL P
    (18,3,RPT$(A$,6))
610 A$=" "&CHR$(131)&CHR$(128)&CHR$
    (132):: CALL P(9,3,RPT$(A$,6))
620 CALL P(19,3,RPT$(A$,6))
630 CALL P(10,5,"f  e  d  c  b
    a"):: CALL P(16,5,"a  b  c
    d  e  f")
640 A$=CHR$(129)&RPT$(CHR$(128),2)&
    CHR$(130):: A$=A$&RPT$(" ",21)&
    A$
650 CALL P(12,1,A$)
660 A$=RPT$(CHR$(128),4):: A$=A$&RP
    T$(" ",21)&A$ :: CALL P(13,1,A$
    )
670 A$=CHR$(131)&RPT$(CHR$(128),2)&
    CHR$(132):: A$=A$&RPT$(" ",21)&
    A$
680 CALL P(14,1,A$):: CALL P(15,1,"
    ")
690 CALL SPRITE(#5,143,5,5,14)
700 CALL HCHAR(12,31,130):: CALL HC
    HAR(13,31,128):: CALL HCHAR(14,
    31,132)
710 FOR I=0 TO 12 :: F(I)=6 :: NEXT
     I ! Anfangsverteilung ******
    ***********
720 F(6),F(13)=0 ! Heimfelder ***
    **********
730 FOR I=0 TO 13 :: IF I=6 OR I=13
     THEN 750
740 CALL SETZ(I,F())
750 NEXT I
760 !
770 ! Spielbeginn ++++++++
780 ENDE,ZA=0 :: L$(N)=""
790 IF RND).5 THEN HF=13 :: L$(N)="
    " :: ZA=1 ELSE HF=6 ! Spieler
    oeffner best.
800 FOR I=22 TO 24 :: CALL HCHAR(I,
    1,30,32):: NEXT I
810 ON INT(HF/6)GOTO 820,1020
820 !
830 ! Spieler waehlt +++++
840 CALL P(22,16,"DEIN ZUG?"):: ZUG
    =1 :: GOTO 860
850 CALL P(24,16,"NOCH MAL?"):: ZUG
    =0 ! 2. Zug *********
860 CALL SOUND(200,1400,5)
870 CALL KII(49,54,K)
880 IF K()83 THEN K=K-49 :: GOTO 90
    0
890 POI(2)=POI(2)+1 :: CALL DELSPRI
    TE(ALL):: GOTO 420 ! Abbruch=N
    iederlage ***
900 IF F(K)=0 THEN 860
910 CALL P(24-2*ZUG,26,CHR$(K+104))
    ! Zuganzeige ******
920 ANZ=F(K)! Anzahl der Kugeln **
    *********
930 GOSUB 1270 ! Zug speichern **
    *********
940 GOSUB 1170 ! Kugel ins Feld le
    gen ****
950 GOSUB 1190 ! Kugeln verteilen
    ********
960 IF F(K)=1 AND K()6 AND K()13 TH
    EN 970 ELSE 980 ! Test auf Gew
    innfeld **
970 IF F(12-K)()0 THEN CALL GEW(HF,
    K,F())
980 GOSUB 1360 ! Test auf Spielend
    e ******
990 IF ENDE=1 THEN 1640
1000 IF K=6 AND ZUG=1 THEN 850 ! T
     est auf 2. Zug *****
1010 HF=13 :: GOTO 800 ! Spielerwe
     chsel ********
1020 !
1030 ! Computer waehlt +++
1040 CALL P(22,4,"ICH ZIEHE@"):: ZU
     G=1 :: GOTO 1060
1050 CALL P(24,5,"NOCH MAL@"):: ZUG
     =0
1060 GOSUB 1410 ! Auswahl des Zuge
     s *******
1070 GOSUB 1270
1080 CALL P(24-2*ZUG,15,CHR$(K+97))
     ! Zuganzeige ******
1090 ANZ=F(K):: GOSUB 1170
1100 GOSUB 1190
1110 IF F(K)=1 AND K()6 AND K()13 T
     HEN 1120 ELSE 1130 ! Test auf
      Gewinnfeld **
1120 IF F(12-K)()0 THEN CALL GEW(HF
     ,K,F())
1130 GOSUB 1360
1140 IF ENDE=1 THEN 1640
1150 IF K=13 AND ZUG=1 THEN 1050 !
      Test auf 2. Zug ***
1160 HF=6 :: GOTO 800 ! Spielerwec
     hsel ********
1170 !
1180 CALL HCHAR(13,8,113,ANZ):: RET
     URN ! Kugeln ins Feld legen
     ***************
1190 !
1200 F(K)=0 :: CALL SETZ(K,F())! K
     ugeln verteilen *************
     ***************
1210 FOR I=ANZ TO 1 STEP -1 :: K=K+
     1 :: IF K=14 THEN K=0
1220 F(K)=F(K)+1 :: IF K=6 OR K=13
     THEN CALL HEIM(K,F()):: GOTO 1
     240
1230 CALL SETZ(K,F())
1240 CALL HCHAR(13,7+I,32):: CALL S
     OUND(150,700,5)! Kugel im Fel
     d loeschen ***
1250 FOR ZT=1 TO 100 :: NEXT ZT
1260 NEXT I :: RETURN
1270 !
1280 JJ=K :: IF JJ)6 THEN JJ=JJ-7 !
      Zug speichern ****
1290 ZA=ZA+1 :: IF ZA(21 THEN L$(N)
     =L$(N)&STR$(JJ)
1300 RETURN
1310 !
1320 ! Verlorene Spiele vergleiche
     n ******************
1330 K=J-7 :: FOR I=0 TO N-1 :: A$=
     SEG$(L$(N),1,ZA)&STR$(K)
1340 IF A$=SEG$(L$(I),1,ZA+1)THEN B
     I=BI-2
1350 NEXT I :: RETURN
1360 !
1370 FOR I=0 TO 5 :: IF F(I)()0 THE
     N 1390 ! Test auf Spiel-Ende
      **************
1380 NEXT I :: ENDE=1 :: RETURN
```

```
1390 FOR I=7 TO 12 :: IF F(I)<>0 TH
     EN RETURN
1400 NEXT I :: ENDE=1 :: RETURN
1410 !
1420 ! Auswahl Computerzug ++++++
     ++++++++++++++++++++
1430 V=-70 :: FOR I=0 TO 13 :: C(I)
     =F(I):: NEXT I ! Verteilung s
     peichern *****
1440 FOR J=7 TO 12 :: IF F(J)=0 THE
     N 1620
1450 JJ=J :: ANZ=F(JJ):: F(JJ)=0
1460 FOR I=ANZ TO 1 STEP -1 :: JJ=J
     J+1 :: IF JJ=14 THEN JJ=0
1470 F(JJ)=F(JJ)+1 :: NEXT I
1480 IF F(JJ)=1 AND JJ<>6 AND JJ<>1
     3 THEN 1490 ELSE 1520
1490 IF F(12-JJ)<>0 THEN F(HF)=F(HF
     )+F(12-JJ)+1 :: F(JJ),F(12-JJ)
     =0
1500 FOR X=0 TO 5 :: IF F(X)<>0 THE
     N 1520 ! Check auf k.o.-Zug
     ***************
1510 NEXT X :: IF F(13)>F(6)THEN K=
     J :: RETURN
1520 ! Check des gegnerischen Zuge
     s ******************
1530 FOR I=0 TO 5 :: IF F(I)=0 THEN
      1580
1540 S=F(I)+I :: U=0
1550 IF S>13 THEN S=S-14 :: U=U+1 :
     : GOTO 1550
1560 IF (F(S)=0)+(F(S)=14)=-1 AND S
     <>6 AND S<>13 THEN U=F(12-S)+U
1570 IF U>DI THEN DI=U
1580 NEXT I
1590 DI=F(13)-F(6)-DI :: IF ZA<20 T
     HEN GOSUB 1310
1600 FOR I=0 TO 13 :: F(I)=C(I):: N
     EXT I ! gespeicherte Verteilun
     g zurueck
1610 IF DI>=V THEN K1=J :: V=DI
1620 NEXT J
1630 K=K1 :: RETURN
1640 !
1650 ! Ende der Runde ++++
1660 IF F(6)>F(13)THEN POI(1)=POI(1
     )+1 :: ST=1 :: CALL GMEL :: GO
     TO 1690
1670 IF F(13)>F(6)THEN POI(2)=POI(2
     )+1 :: ST=2 :: CALL VMEL :: GO
     TO 1690
1680 UN=UN+1 :: ST=3 :: CALL UMEL
1690 CALL DELSPRITE(ALL):: CALL CLE
     AR :: CALL COL(2,8,4)
1700 CALL P(2,9,"AUSWERTUNG"):: CAL
     L P(3,9,RPT$(CHR$(142),10))
1710 CALL P(6,1,"ERGEBNIS DES LETZT
     EN SPIELS:")
1720 ON ST GOTO 1730,1740,1750
1730 N=N+1 :: CALL P(8,1,STR$(F(6))
     &":"&STR$(F(13))&" FüR DICH !"
     ):: GOTO 1760
1740 CALL P(8,1,STR$(F(13))&":"&STR
     $(F(6))&" FüR MICH !"):: GOTO
     1760
1750 N=N+1 :: CALL P(8,1,"UNENTSCHI
     EDEN.")
1760 SC(1)=SC(1)+F(6):: SC(2)=SC(2)
     +F(13)
1770 DISPLAY AT(14,1):USING "SCORE
     C ####:":SC(2)
1780 CALL P(14,18,STR$(SC(1))):: CA
     LL P(14,23,"S")
1790 CALL P(18,10,"SIEG MOLG UN PKT
     E"):: CALL P(20,1,"COMPUTER")
1800 CALL P(22,1,"SPIELER")
1810 DISPLAY AT(20,11):USING "##
     ##  ## ##:":POI(2),POI(1),UN,P
     OI(2)*2+UN
1820 CALL P(20,26,STR$(POI(1)*2+UN)
     )
1830 DISPLAY AT(22,11):USING "##
     ##  ## ##:":POI(1),POI(2),UN,P
     OI(1)*2+UN
1840 CALL P(22,26,STR$(POI(2)*2+UN)
     )
1850 CALL KEY(0,K,S):: ZT=ZT+1 :: I
     F S=0 THEN ZT=ZT+1
1860 IF S=0 AND ZT<2500 THEN 1850 E
     LSE ZT=0 :: GOTO 420
1870 !
1880 ! Ende +++++++++++++
1890 CALL CLEAR :: END
1900 !
1910 ! Spielregeln +++++++
1920 CALL CLEAR :: CALL COL(2,4,8)
1930 CALL P(1,8,"SPIELREGELN:"):: C
     ALL P(2,8,RPT$(CHR$(142),12))
1940 CALL P(4,1,"AWARI IST EIN SPIE
     L, DAS MIT7 HöLZERN UND 72 STE
     INEN GE-")
1950 CALL P(6,1,"SPIELT WIRD, DIE F
     OLGENDER- MASSEN AUSGELEGT WER
     DEN:")
1960 CALL P(9,10,"TI-SEITE"):: CALL
      P(11,4,"6  5  4  3  2  1
     ")
1970 B$=CHR$(140)&"000" :: B$="  "&B
     P1$(B$,6)&CHR$(140):: CALL P(1
     2,1,B$)
1980 CALL P(13,1,B$):: CALL P(15,1,
     B$):: CALL P(16,1,B$)
1990 B$=CHR$(140)&"  " :: CALL P(1
     4,1,"I"&RPT$(B$,6)&CHR$(140)&"
     HF")
2000 CALL P(17,4,"1  2  3  4  5
     6"):: CALL P(19,8,"DEINE SE
     ITE")
2010 CALL P(22,1,"'I'  = TI-HEIMFEL
     D"):: CALL P(23,1,"'HF' = DEIN
      HEIMFELD")
2020 CALL P(24,27,"))"):: CALL KII(
     13,127,K):: CALL CLEAR
2030 CALL P(2,1,"JEDER SPIELER BESI
     TZT ALSO  6 FELDER UND 1 'HEIM
     FELD'.")
2040 CALL P(5,1,"EIN ZUG BESTEHT DA
     RIN,")
2050 CALL P(7,1,"1. ALLE STEINE AUS
      EINEM")
2060 CALL P(8,1,"  EIGENEN FELD ZU
      ENTNEHMEN  UND")
2070 CALL P(11,1,"2. DIESE DANN ENT
     GEGEN DEM")
2080 CALL P(12,1,"  UHRZEIGERSINN
     IN DIE FOL-  GENDEN FELDER ZU
     SETZEN.")
2090 CALL P(15,1,"LANDET DER ZULETZ
     T GESETZTE STEIN IN EINEM LEER
     EN FELD,")
2100 CALL P(17,1,"WAHREND DAS GEGEN
     UBERLIEGEN-DE BESETZT IST, SO
     WERDEN")
2110 CALL P(19,1,"SAMTLICHE STEINE
     DIESER BEI-DEN FELDER EROBERT
     UND LAH-")
2120 CALL P(21,1,"DEN IN DEN HEIMFE
     LD DES AM  ZUG BEFINDLICHEN SP
     IELERS.")
2130 CALL P(24,27,"))"):: CALL KII(
     13,127,K):: CALL CLEAR
2140 CALL P(2,1,"WIRD DER LETZTE ST
     EIN EINES SPIELZUGS IN EIGENEN
     HEIM-")
2150 CALL P(4,1,"FELD ABGELEGT, SO
     ERHALT DERSPIELER EINEN ZWEITE
     N ZUG.")
2160 CALL P(7,1,"DAS SPIEL IST BEEN
     DET, WENN EINE DER BEIDEN SEIT
     EN")
2170 CALL P(9,1,"LEER IST."):: CALL
      P(11,1,"SIEGER IST DER SPIELE
     R,")
2180 CALL P(12,1,"DESSEN HEIMFELD D
     IE GROSSEREANZAHL AN STEINEN A
     UFWEIST.")
2190 CALL P(15,1,"DAS ZUFALLS-'LOS'
      BESTIMMT  DEN ERöFFNENDEN SPI
     ELER.")
2200 CALL P(18,1,"***  DAS SPIEL KA
     NN MIT ******  DER TASTE 'S'
     JE-  ***")
2210 CALL P(20,1,"***  DERZEIT ABGE
     BROCHEN ******  WERDEN.
     ***")
2220 CALL P(23,5,"V I E L  S P A S
     S"):: CALL P(24,27,"))")
2230 CALL KII(13,127,K):: GOTO 420
2240 !
2250 !
2260 SUB P(X,Y,B$)! Display at **
     **********************
2270 DISPLAY AT(X,Y):B$ :: SUBEND
2280 !
2290 SUB COL(X,Y,Z)! Farben defini
     eren ***************
2300 CALL SCREEN(Z)
2310 FOR I=1 TO 14 :: CALL COLOR(I,
     X,Y):: NEXT I :: SUBEND
2320 !
2330 SUB KII(X,Y,K)! Call key ***
     **********************
2340 CALL KEY(U,K,S):: IF S<>0 THEN
      2340
2350 IF K=83 THEN SUBEXIT
2360 IF K<X OR K>Y THEN 2340
2370 SUBEND
2380 !
2390 SUB GEW(K,Y,F())! Gewinn addi
     eren ***************
2400 F(K)=F(K)+F(12-Y)+1 :: CALL HE
     IN(K,F())
2410 CALL SOUND(100,300,5):: CALL S
     OUND(200,600,3):: F(Y),F(12-Y)
     =0
2420 CALL SETZ(Y,F()):: CALL SETZ(1
     2-Y,F()):: SUBEND
2430 !
2440 SUB SETZ(U,F())! Setzen ****
     **********************
2450 IF U<6 THEN X=18 :: Y=6 :: V=1
      ELSE X=8 :: Y=26 :: V=-1
2460 J=U :: IF J>6 THEN J=J-7
2470 IF F(U)=0 THEN CALL HCHAR(X,Y+
     (V*4*J),128,2):: SUBEXIT
2480 S=F(U):: IF S<10 THEN 2500
2490 CALL HCHAR(X,Y+(V*4*J),ASC(STR
     $(INT(S)/10)+"0")):: S=S-10*INT(S/
     10)
2500 CALL HCHAR(X,Y+1+(V*4*J),ASC(S
     TR$(S)+"0")):: SUBEND
2510 !
2520 SUB GMEL ! Gewinnmelodie ***
     **********************
2530 FOR T=1 TO 15 :: CALL SOUND(-1
     00,220,0):: CALL SOUND(-100,44
     0,0):: NEXT T
2540 SUBEND
2550 !
2560 SUB VMEL ! Verliererweledie
     **********************
2570 FOR I=20 TO 5 STEP -5 :: FOR J
     =I TO 1 STEP -1
2580 CALL SOUND(-100,110+J*8,2):: N
     EXT J :: NEXT I :: SUBEND
2590 !
2600 SUB UMEL ! Unentschieden ***
     **********************
2610 FOR I=1 TO 3 :: CALL SOUND(200
     ,200,3):: CALL SOUND(300,240,3
     ):: NEXT I
2620 SUBEND
2630 !
2640 SUB HEIN(K,F())! Heimfeldanze
     ige ****************
2650 IF K=13 THEN Y=3 :: P=21 ELSE
     Y=1 :: P=221
2660 I=F(K):: IF I<10 THEN 2680
2670 CALL SPRITE(#Y,ASC(STR$(INT(I)
     /10)),16,92,P):: I=I-10*INT(I/
     10)
2680 CALL SPRITE(#Y+1,ASC(STR$(I)),
     16,92,P+12):: SUBEND
```

**Listing zu »Awari«**

des Spielfeldes gesetzt und dann auf die folgenden Felder aufgeteilt.

In der Eröffnungspartie spielt der Computer noch standardmäßig, da er zu wenige »Erkenntnisse« über seinen Gegner besitzt. Dies ändert sich jedoch mit jeder Verlustpartie. Um Kopien dieser Runden zu vermeiden, werden für Züge, die zu gleichem Spielverlauf führen, Alternativen gesucht.

Bei Erreichen des Spielendes wird zunächst der Ausgang der Partie durch eines der drei Unterprogramme GMEL, VMEL oder UMEL akustisch untermalt. Im Anschluß daran wird der aktuelle Spielstand in Form einer kleinen Tabelle ausgegeben (Zeile 1640 bis 1860). Das Betätigen einer beliebigen Taste oder auch einige Momente geduldigen Wartens, führen zurück zum Spielmenü.

Ein vorzeitiger Abbruch der Partie kann im übrigen mit Hilfe der Taste »S« erreicht werden, was allerdings als Niederlage gewertet wird. (Hans Jürgen Adler)

## Atari-Listings gesucht

Wir freuen uns über die rege Beteiligung an unserem Wettbewerb »Listing des Monats«. Leider sind unter den vielen Zusendungen nur sehr wenige Atari-Programme. Deshalb möchten wir heute besonders alle Atari-Besitzer bitten, ihr Heimarchiv zu durchwühlen. Vielleicht ist das eine oder andere Programm auch für unsere Leser interessant. Selbstverständlich werden alle abgedruckten Listings honoriert. Nicht vergessen: Alle Zusendungen nehmen an der Aktion »Listing des Monats« teil (Gewinnchance 2000 Mark). Ist das nicht einen Versuch wert? Richten Sie Ihre Zusendungen bitte an

**Redaktion Happy Computer, Werner Breuer, Hans-Pinsel-Str. 2, 8013 Haar bei München.**

| Zeile | Bedeutung |
|---|---|
| 1-9 | REM-Zeilen (können auch gelöscht werden) |
| 10 | Aufruf des Anfangsbildes |
| 11 | Aufruf des Begrüßungsbildes |
| 12 | Aufruf einer Routine, die ein Muster auf den Bildschirm schreibt |
| 13 | Home-Zeichen. Setzt man in den String auch eine Farbumschaltung, kann man die Farbe des Laufschrifttextes umändern |
| 14-21 | REM-Zeilen (können auch gelöscht werden) |
| 22-24 | Laufschrifttext (in A$) |
| 25 | Bildschirm wird eingestellt (Bildschirmfenster wird in X-Richtung verkleinert, Bildschirmfarben umgeändert...) |
| 26 | Textverschieberoutine (im Buchstabenraster) |
| 27 | Textverschieberoutine (im Bildschirmpunktraster) |
| 30 | Warteschleife |
| 31-35 | schreibt ein Muster auf den Bildschirm |
| 36 | Verhindert Flackern des Textes, indem das Verschiebezählerrücksetzen mit dem Rastertakt synchronisiert wird (Schaltet um, wenn sich der Rasterpunkt in der unteren Hälfte des Bildschirms befindet) |
| 37 | Schreibt Text in Verschiebung eines Buchstabens und setzt Verschiebezähler auf 7 |
| 38 | Initialisiert den Bildschirm für das Begrüßungsbild |
| 39-48 | Schreibt Begrüßungsbild |
| 49-52 | REM-Zeilen (können auch gelöscht werden) |
| 53-59 | Schreibt Anfangsbild und läßt schwarze Balken über den Bildschirm laufen |

**Aufschlüsselung nach Zeilennummern**

gehen, damit kein Flackern entsteht.

Das Verschieben um einen Bildschirmpunkt besorgt Zeile 27. Das Rücksetzen des Verschiebezählers (in diesem Falle Rückkehr von 1 nach 7) und das Weiterbewegen des Textes (nächster Buchstabe) macht Zeile 37 beziehungsweise Zeile 26.

Den Text in A$ (Zeilen 22 bis 24) ändern Sie in einen beliebigen Text um. Beachten Sie jedoch, daß vor dem Textbeginn und nach dem Textschluß mindestens 40 Spaces (Leerstellen) sein müssen. Außerdem darf die Gesamtlänge (einschließlich Leerstellen) von A$ nicht größer als 255 sein. Sollten Sie die Länge von A$ umändern, müssen Sie ebenso die Zahl 116 in Zeile 26 ändern.

**Zum Listing**

Bevor Sie das Programm eingeben, schalten Sie auf den Klein/Großschriftmodus um, da im Programm Klein- und Großbuchstaben verwendet werden.

Das »Kreuz mit einem weißen Punkt in der Mitte« in Zeile 31 und 35 entspricht der Funktion Cursor yellow (CTRL+8). Das Zeichen vor »hg-software systems« (Zeile 47) ist die Farbumschaltung auf schwarz (CTRL+1).

Alle sonstigen Farbumstellungen dürften zu identifizieren sein.

(Holger Gehrmann)

# Ungewöhnliche Edit-Erweiterung: Help

**Wer hat sich nicht schon mal einen etwas intelligenteren Computer gewünscht, wenn in Programmen Fehler zu suchen waren und der Maschine nur magere Fehlermeldungen von der Sorte »SYNTAX ERROR« zu entlocken waren. Eine ganz besondere Edit-Erweiterung für den TI 99/4A (mit Extended Basic Modul) verleiht diesem aber fast so etwas wie Intelligenz.**

Betriebssystemerweiterungen sind bei allen Computern sogar für ausgefuchste Programmierer eine schwierige Sache. Beim TI 99/4A kommt noch die reichlich undurchsichtige innere Struktur hinzu. Deshalb bedarf es im Fall der vorliegenden Erweiterung größter Sorgfalt bei der Eingabe des Programms. Es werden Systemadressen verwendet, die selbst im Handbuch zum Editor/Assembler nicht erwähnt wurden. Eine Erklärung der einzelnen Schritte würde den Rahmen dieser Beschreibung leider sprengen. Deshalb beschränkt sich der Text nur auf die Erläuterung des Eingabe- und Anwendungsmodus.

Sinn des Programms ist es, im Edit-Modus eine zusätzliche Hilfsfunktion zu bieten, die mit dem Befehl »HELP« aufgerufen wird. Sie hat allerdings nur Sinn, wenn vorher eine Fehlermeldung am Schirm erschienen ist, wie zum Beispiel »SYNTAX ERROR«. In diesem Fall erscheint ein Ausgabetext (in der Zeile 650 zu sehen), zusammen mit dem richtigen Token (im Falle des SYNTAX ERRORS) oder einem numerischen Wert, je nach Fehler.

Das eigentliche Geheimnis dieser Zusatzfunktion ist natürlich in den Data-Zeilen mit den Codes (Zeile 570 bis 610) verborgen. Da durch die komplizierte interne Struktur des TI 99/4A falsche Poke-Werte in einigen der verwendeten Adressen länger anhaltende Veränderungen des eingebauten Basics hervorrufen können (weil Treiber gegeneinander arbeiten und dadurch ROM-Zellen aufladen kön-

# »Help«

TI 99/4A

nen), sollte das sehr empfindliche Maschinenprogramm vor einem Probelauf zur Sicherheit unbedingt gesondert auf Richtigkeit der Werte überprüft werden. Dazu dienen die Zeilen 700 bis 760.

Bewährt hat sich folgende Reihenfolge: Erst die Zeilen 570 bis 760 eingeben, mit »RUN 750« starten und die »OK«-Meldung abwarten. Wenn die Nachricht »INKORREKT« erscheint, müssen unbedingt die Data-Zeilen und der Referenzwert in Zeile 760 überprüft werden. Wenn der Fehler gefunden wurde, darf erneut mit »RUN 750« gestartet werden. Erst wenn bei einem Testlauf das »OK« erscheint, können alle Zeilen ab 680 gelöscht werden. Nun sind die restlichen Zeilen einzugeben.

Einzelne Probeläufe von Programmteilen dürfen unter keinen Umständen stattfinden. Die Gefahr falscher Pokes und unkontrollierte Vektoränderungen ist zu groß. Sichern Sie aber zwischendurch ihre Programmzeilen auf Diskette oder Kassette. Bevor Sie die Eingabe beenden, müssen alle Zeilen im »Trockengang« — also ohne Zwischenstarts — auf genaue Übereinstimmung mit dem Listing überprüft werden. Sichern Sie auch diese endgültige Version erst, bevor Sie weitermachen.

Jetzt müssen zwei verschiedene Wege eingeschlagen werden, je nachdem, ob eine Diskettenstation oder nur ein Kassettenrecorder zur Verfügung steht. Haben Sie eine Diskettenstation, dann gehen Sie am besten so vor (wichtig ist die Reihenfolge!): In Laufwerk 1 kommt die Diskette, mit der bei Arbeitsbeginn die Erweiterung zum System geladen werden soll. Die Abspeicherung erfolgt mit »SAVE DSK1.LOAD«. Anschließend kann »FCTN =« gedrückt werden. Dieser Reset bringt das Titelbild auf den Schirm. Jetzt sind eine beliebige Taste und anschließend die »2« für »Extended Basic« zu drücken. Die Erweiterung wird automatisch von der Diskette in den Computer geladen. In Zukunft braucht nur noch die Diskette nach Erscheinen des Eingangsbildes eingelegt zu werden.

Wichtig: Das Programm darf man unter keinen Umständen mit »OLD DSK1.LOAD« manuell laden und starten, sonst kann es zerstört werden, da der manuelle Zugriff einige Daten im Programm verändert. Aus dem gleichen Grund darf das Programm auch nicht unter einem anderen Namen als »LOAD« gestartet werden!

Bei Verwendung eines Kassettenrecorders ist die Sache einfacher. Die Abspeicherung geschieht mit »SAVE CS1«. Laden darf man es nur mit der Befehlsfolge »RUN "CS1.HELP"« (Autostart)! Sobald man das Programm mit »OLD CS1« in den Computer holt und manuell startet, werden auch hier unter Umständen wichtige Programmteile zerstört.

Ein kluger Mann soll einmal gesagt haben, wenn Maschinen wirklich Intelligenz besäßen, würden sie aufhören für uns zu arbeiten. Aber das war wohl lange vor dem Zeitalter der Computer. (A. Ripl)

```
100 !******************
110 !*OOPS-ERWEITERUNG*
120 !******************
130 !(C)HAPPY-COMPUTER
140 !    1.4.1983
150 !
160 !
170 !  FUER TI99/4A
180 !
190 ON BREAK NEXT
200 CALL CLEAR :: CALL SCREEN
    (8):: ON WARNING NEXT
210 !-----------------------
220 !AUSGABESTRING BILDEN
230 !-----------------------
240 RANDOMIZE 37657 :: CALL I
    NIT :: CALL PEEK(2454,X):
    : CALL LOAD(3*818,X)
250 FOR A=1 TO 5 :: P$(A)=""
    :: READ L
260 FOR B=1 TO L :: READ PDAT
270 P$(A)=P$(A)&CHR$(PDAT/INT
    (RND*8+1))
280 NEXT B :: FG=-1 :: NEXT A
290 FOR A=1 TO 5 :: READ HILF
    $(A):: NEXT A
300 !-----------------------
310 !SYSTEM-EINGABEN ABFANGEN
320 !-----------------------
330 GOSUB 660
340 ON ERROR 670
350 CALL VCHAR(24,2,0*30345+(
    2*(N+1))*31):: CALL SOUND
    (200,30000,30)
360 INPUT "":E$
370 IF E$="LIST" THEN 490
380 IF E$="RUN" THEN 510
390 IF E$="HELP" THEN 520
400 IF ASC(E$)<48 OR ASC(E$))
    57 THEN 530 ELSE 480
410 !-----------------------
420 !HILFSTEXT AUSGEBEN
430 !-----------------------
440 CALL PEEK(245,X):: PRINT
    :HILF$(F+N*X):: PRINT
450 IF F<>2 THEN 32767
460 FOR A=1 TO 200 :: NEXT A
    :: CALL CLEAR :: CALL SCR
    EEN(15+Q*XX)
470 CALL SOUND(-1000,5*F6,1):
    : FOR A=1 TO 200 :: NEXT
    A :: GOTO 470
480 CALL PEEK(236,X):: GOTO 3
    50
490 CALL PEEK(1035,X):: F=X*3
    *N+1 :: GOTO 440
500 GOTO 350
510 CALL PEEK(1037,X):: F=X*3
    *N+2 :: GOTO 440
520 CALL PEEK(1039,X):: F=X*3
    *N+4 :: GOTO 440
530 CALL PEEK(1041,X):: F=X*3
    *N+3 :: GOTO 440
540 !-----------------------
550 !MASCHINEN-CODES
560 !-----------------------
570 DATA 41,42,256,365,536,57
    6,64,308,414,164,150,234,
    539,146,374,96,136,553,40
    2,432,128,312,292,536,432
    ,672,192
580 DATA 128,160,32,96,148,34
    5,340,138,312,224,77,219,
    249,420,165,28,168,160,16
    4,69,156,546,256,476,316,
    335,216
590 DATA 192,83,138,304,330,1
    36,164,220,96,168,581,469
    ,144,340,552,664,33,14,29
    4,192,581,356,234,420,130
    ,616
600 DATA 224,207,82,410,237,2
    46,18,84,96,360,365,456,7
    0,192,272,365,328,32,249,
    69,228,264,249,588,132
610 DATA 9,252,224,328,345,39
    0,340,445,256,42
620 !-----------------------
630 !AUSGABE-TEXTE
640 !-----------------------
650 DATA * SO GEHT ES WEITER:
    ,* EINZUGEBEN IST:,* FEHL
    ER BEI ZEICHEN-NR.:,* RIC
    HTIG IST:,* WEITER AB:
660 FOR Z=1 TO (N+1)*5 :: HIL
    F$(Z)=P$(Z):: NEXT Z :: P
    RINT :: PRINT P$(5):: PRI
    NT :: RETURN
670 RETURN 340
680 !
690 !
700 !************************
710 !DIE FOLGENDEN ZEILEN UEB
    ERPRUEFEN DIE DATAZEILEN
    AUF FEHLER. VOR PROBE-LAU
    F
720 !HIT))RUN 750((STARTEN!
730 !SIE KOENNEN NACH PRUEFUN
    G GELOESCHT WERDEN!
740 !************************
750 FOR A=1 TO 115 :: READ B
    :: C=C+B :: NEXT A
760 IF C=30442 THEN PRINT "OK
    " ELSE PRINT "INKORREKT"
```

**Listing zur Erweiterung des Betriebssystems (»HELP«)**

## Joystick-Kommentare

Der von Ihnen hoch eingestufte Atari-Joystick (Stabilität: 2) hat bei uns nun bereits zum zweiten Male Bruch gemacht, das heißt der Druckring (dünnes Plastik) des eigentlichen Steuerknüppels ist gebrochen. Einzige Alternative nach diesem Bruch ist der Kauf eines neuen Joysticks, da nach Anfragen in mehreren Fachgeschäften und bei Atari keine Ersatzlieferung möglich ist. Leider wurde dieses Problem bei Ihrem Test nicht berücksichtigt. Nach Aussagen des Fachhandels (hier MCC-Kiel) wird dieser Bruch laufend gemeldet.

Wolfgang Miensopust

Der Atari-Trakball ist übrigens kompatibel zu den meisten (laut Verpackung sogar zu allen!) Spielen auf dem Commodore 64. Es gibt nämlich Gerüchte, daß dieser Trakball nur bei Paddles-Spielen laufen soll, da nur die Paddles den im 64er eingebauten A-D-Wandler benutzen, die Joysticks aber nicht. Deshalb können Paddles-Spiele nicht mit einem Joystick gespielt werden und umgekehrt. Joystickspiele laufen aber mit dem Atari-Trakball, sogar der Koala-Painter kann ohne das Koala-Pad mit dem Trakball sehr gut bedient werden!

Es gibt noch mehr sehr interessante Joysticks, die nicht vorgestellt wurden:

— Von Wico gibt es in Amerika noch einen Joystick namens Wico The Source. Dieser Joystick arbeitet mit analogen Potentiometern und kann deshalb an Apple, IBM und Radio Shack (Tandy)-Rechnern benutzt werden. Dieser Joystick wird nicht nur für Arcade-Spieler interessant sein, sondern auch für kommerzielle Nutzung ist dieser Joystick empfehlenswert, da die Potentiometer eine sehr genaue Steuerung ermöglichen.

Die Firma TG Products (in Texas) bietet einen sehr guten Joystick an, der sowohl für Links- als auch für Rechtshänder gleichermaßen geeignet ist: TG EnJOYstick. Der Joystick liegt in der Handfläche, man steuert den Hebel, der positiv kurz ist, mit der anderen Hand. Der Button ist direkt am Geäuse befestigt (also nicht am Griff) und wird vom Daumen jener Hand bedient, die den Joystick hält. Der Joystick kann deshalb auch von Linkshändern bedient werden, weil der vordere Teil des Gehäuses mit dem Button herausgenommen und um 180° gedreht werden kann, der Button also je nach Wunsch links oder rechts am Gehäuse ist. Der Joystick läuft am Atari und Commodore 64/20.

Neben dem Joystick gibt es in den USA noch einige andere interessante Eingabegeräte für VC 20 und Commodore 64. So bieten verschiedene Firmen Zehner-Tastaturen an: Cardco (Wichita, Kansas) bietet einen 10er-Block mit 16 Tasten an, die auch einzeln programmiert werden können. Auch eine Taschenrechner-Simulation ist möglich. Diese Tastatur wird einfach am Joyport eingesteckt. Kosten: 39,95 Dollar. Computor Place (Torrance, CA) bietet ebenfalls eine 10er-Tastatur an, die 69,95 Dollar kostet. Es ist schade, daß Sie in Ihrer »Controller«-Reihe die Lightpens nicht berücksichtigt haben, denn auch dieses Eingabemedium hat seine Freunde. Es gibt in Deutschland einige Firmen, die Lightpens anbieten, aber der meines Wissens beste, sprich einwandfrei funktionierende Joystick kommt wieder aus den USA: Edumate Light Pen von Futurehouse (Chapel Hill, NC). Dieser Lightpen wird für Atari, Commodore 64 und VC 20 angeboten und inklusive Software für 29,95 Dollar verkauft.

Bei den Berichten über das Koala-Pad (Koala-Painter) haben Sie bereits erwähnt, daß Koala Technologies Corporation plane, weitere Programme für das Koala-Pad herauszubringen. In den USA ist dies inzwischen schon geschehen:

— Logo Design Master
— Dancing Bear
— Spider Eater
— Spellicopter
— Coloring Book
— Mocroillustrator
— The Illustrator

Ich weiß allerdings nicht, ob für alle diese Programme unbedingt das Pad notwendig ist.

Detlef Wacker

## Atari XL nicht beachtet?

***Bei den Atari-Spiele-Tests, die zwar recht objektiv sind, werden die neuen XL-Modelle gar nicht beachtet. Existieren diese Computer für Sie gar nicht?***

***H. Vicher***

Die meisten für Atari 400/800 angebotenen Spiele laufen auch auf A 600/800 XL, insbesondere die neuen Spiele, die wir in erster Linie testen.

## 99/4A-Tip

Seit einem Jahr gibt es das Mini-Memory für den TI 99/4A. Es erlaubt außer dem Zugriff auf Maschinensprache und der Benutzung als schnelles, batteriegepuffertes Speichermedium auch den Zugriff auf Systemspeicherstellen. Ein kleiner Tip:

```
CALL PEEK (—31952,A,B)
—ENTER—
PRINT 16384-256*A-B
—ENTER—
```

Mit dieser Prozedur erhält man den von einem Programm belegten Speicherraum.

Ralf Biedermann

## VC 20-Tip

Als begeisterter VC 20-User möchte ich für einige Probleme, die speziell bei Commodore auftreten, Lösungswege aufzeigen. Dem Einsteiger, der noch größtenteils fertige Programme abtippt, stellt sich meist das Problem, daß durch ungenügende Druckqualität der Listings, vor allem die Steuerzeichen kaum noch erkennbar sind. Ich habe es mir daher zur Gewohnheit gemacht, meine Programme so zu schreiben, daß diese Schwierigkeiten kaum noch auftreten können. Nehmen wir als Beispiel eine Druckanweisung in die 18. Zeile. Normalerweise steht sie in folgendem Format im Programm:

```
.
100 PRINT" QQQQQQQQ
QQQQQQQQQQHALLO HIER
BIN ICH"
.
```

(S = Cursor Home ; Q = Cursor Down)

Nun ist aber das Abzählen der Steuerzeichen keine so richtige faszinierende Beschäftigung, die mich unbedingt begeistert. Es geht auch anders:

```
.
10 DO$ = CHR$(19) : REM CUR-
SOR HOME
12 FORT = 1 TO 22 : DO$ =
DO$ + "Q" : REM CURSOR
DOWN
14 NEXT T
.
100 PRINT LEFT$(DO$,19)"
HALLO HIER BIN ICH"
.
```

Diese Version ist zwar aufwendiger in der Programmierung, dafür aber doch sehr viel besser lesbar. Außerdem kann nun mit der LEFT$-Funktion in jede beliebige Zeile gedruckt werden (gewünschte Zeile + 1). Schon bei zweimaligem Anwenden dieser Funktion ist der anfängliche Nachteil der aufwendigeren Programmierung aufgehoben.

Nun noch einige Tips zum INPUT-Befehl:

```
.
100 INPUT" NAME "; NA$
.
```

Drücken Sie bei dieser Version die Return-Taste, bevor eine Eingabe gemacht wurde, dann setzt das Programm in die Variable NA$ einen Leerstring (also nichts). Daher ist unbedingt eine Kontrolle über den Input nötig. Zwei Möglichkeiten:

```
.                    .
100 INPUT" NAME "; NA$
110 IF NA$ = "" THEN 100
.
```

oder noch eleganter, man spricht die Tastatur als Peripherie an:

```
.
100 PRINT" NAME ";: OPEN 1,0
: INPUT 1 , NA$
110 CLOSE 1
.
```

Gerhard Wetz

## Simon's Basic besser erklärt

In Heft 2/84 fragte U. Wendt nach Befehlen, die im Handbuch zu Simon's Basic nicht erklärt sind. Ich habe ein wenig herumprobiert und kam zu folgendem Ergebnis:

SOUND und GRAPHICS sind Konstanten, wohin SOUND = 54272 (=SID Basicadresse) und GRAPHICS = 53248 (= VIC-Basisadresse) sind. Der Befehl NRM schaltet auf Groß-Grafik-Modus um. BCKGNDS a,b,c,d setzt bestimmte Farben:

a: Hintergrundfarbe
b: Untergrundfarbe der Shift-Zeichen
c: Untergrundfarbe der Reverszeichen (und damit auch die Cursorfarbe, nicht zu verwechseln mit der Schriftfarbe)
d: Untergrundfarbe
S H I F T + R V S = Zeichen

BCKGNDS schaltet außerdem auf irgendeinen obskuren Grafik-Modus um (man versuche dann einmal C = — und irgendeine Buchstabentaste zu drücken, bei mir kommt irgendetwas Seltsames heraus). Mit NRM läßt sich das Ganze wieder rückgängig machen.

Lennart Koschella

***Wer kann mir die Bedeutung der Befehle RESUME, RERR, ON TI und DISABLE bei Simon's Basic erklären? Wie finde ich die Plot-Routine?***

***Stephan Sacher***

# pool-

**Nicht jeder hat genügend Platz in seiner Wohnung um einen Pool-Tisch aufzustellen. Mit dem nachfolgenden Programm, für die Grundversion des TI 99/4A, brauchen Sie nicht mehr unbedingt in Spielhallen zu gehen, um Ihr Können zu beweisen.**

Eine Ähnlichkeit des hier vorgestellten Spiels mit dem Original-Pool-Billard, besteht eigentlich nur darin, daß man auch hier die Kugel in ein Loch dirigieren muß. Einen besonderen Reiz bietet das Programm »Pool-Billard«, weil die Abschußposition und das Loch in das die Kugel fallen muß, in jeder Runde eine andere Position einnehmen.

Die senkrechte Markierung, die für einen Augenblick erscheint, gibt an, aus welcher Entfernung die Kugel abgeschossen wird. Die Abschußhöhe kann durch die Eingabe einer Zeilennummer, die auf der linken Bildschirmhälfte angezeigt wird.

**Variablendefinition**

| | |
|---|---|
| A 1 $– A 4 $ | Name der Spieler |
| B | Anzahl der Spieler |
| C | Position vom Loch |
| F – F 7 | Zähler Fehl |
| G – G 3 | Gesamt (Treffer + Fehl) |
| I | Warteschleifen |
| L – L 1 | Abstoßposition |
| Q | Für Zähler Endroutine |
| R | Für Zähler Zeichensatz |
| S – S 2 | Für Zähler Graphik |
| T – T 7 | Zähler Treffer |
| U | Eingabe Zeilennummer |
| X | Ballposition Spalte |
| Y | Ballposition Zeile |
| Z | Für Schlußanzeige |

**Programmbeschreibung zu »Pool-Billard«**

| Zeilennummer | Erläuterungen |
|---|---|
| 10 - 50 | Farbe setzen |
| 60 - 100 | Abfrage ob Spielanleitung |
| 110 - 249 | Spielanleitung |
| 280 - 480 | Eingabe, wieviele Teilnehmer mitspielen |
| 490 - 520 | Eingabe der Namen |
| 530 - 720 | Routine zur Anzeige, welcher Spieler dran ist |
| 730 - 1060 | Hilfsroutine für Grafik |
| 1070 - 1090 | Zufallsgenerator für das Loch |
| 1100 - 1170 | Hilfsroutine für Grafik |
| 1180 - 1210 | Zufallsgenerator für Abstoßposition |
| 1216 - 1251 | Setzen und Löschen der Abstoßposition |
| 1260 - 1280 | Eingabe Zeilennummer |
| 1300 - 1330 | Berechnung der Kugelposition |
| 1340 - 1400 | Test, ob Kugel noch innerhalb der Bande ist |
| 1410 - 1480 | Ton, wenn die Kugel an die Bande kommt |
| 1490 - 1520 | Abfrage, wieviele Teilnehmer |
| 1530 - 1540 | Berechnung Treffer oder Fehl Spieler 4x |
| 1550 - 1650 | Zähler Fehl Spieler 4x |
| 1660 - 1760 | Zähler Treffer Spieler 4x |
| 1790 - 2020 | Zählroutine Spieler 3 |
| 2060 - 2290 | Zählroutine Spieler 2 |
| 2330 - 2560 | Zählroutine Spieler 1 |
| 2570 - 2680 | Zähler Endroutine |
| 2690 - 2790 | Anzeige Treffer und Fehl |
| 2800 - 2880 | Abfrage ob noch ein Spiel |
| 2890 - 3090 | Unterprogramm, Rücksetzen der Zählvariablen auf Null |

TI 99/4A

```
1 REM
2 REM LOTHAR HOFFMANN
3 REM
4 REM ROTEKREUZSTR.26
5 REM
6 REM 3000 HANNOVER 61
7 REM
8 REM        1984
9 REM
10 CALL CLEAR
20 CALL SCREEN(13)
30 FOR R=1 TO 9
40 CALL COLOR(R,2,13)
50 NEXT R
60 PRINT "BENOETIGEN SIE EINE":
   :"SPIELANLEITUNG?": : :
70 PRINT TAB(7);"(J/N)"
80 CALL KEY(0,K,S)
90 IF S=0 THEN 80
100 IF K=ASC("N")THEN 280
110 CALL CLEAR
120 PRINT "**********************
    *******"
130 PRINT "******* POOL-BILLARD
    *******"
140 PRINT "**********************
    *******"
150 FOR I=1 TO 500
160 NEXT I
170 CALL CLEAR
180 PRINT "**********************
    *******": :
200 PRINT "DIESES SPIEL IST FUER
     1-4":"TEILNEHMER GEDACHT.":
     :
210 PRINT "IN DER RECHTEN SPIELF
    LAECHE":"ERSCHEINT FUER CA.
    2.SEC.":"EINE MARKIERUNG.":
    :
218 PRINT "DIESE SENKRECHTE LINI
    E MUSS":"MAN SICH GUT MERKEN
    ,":"DENN VON DIESER POSITION
    ":
220 PRINT "WIRD DIE KUGEL NACH L
    INKS":"OBEN IM WINKEL VON 45
    ' ABGE-":"STOSSEN.": :"AN DE
    R LINKEN SPIELFLAECHE":
221 PRINT "SIND DIE ZEILENNUMERN
     4-21":"GESCHRIEBEN.": :
222 PRINT "**DRUECKEN SIE EINE T
    ASTE**"
225 CALL KEY(0,K,S)
```

# billard

```
226 IF S=0 THEN 225
227 IF K>0 THEN 228
228 CALL CLEAR
229 PRINT "**********************
    *******": :
230 PRINT "NACH DER ABFRAGE(ZEIL
    EN-NR.)":"BESTIMMEN SIE DURC
    H DIE EIN-":"GABE EINER ZAHL
    (4-21),DANN":
231 PRINT "ENTER,VON WELCHER HOE
    HE DIE":"KUGEL ABGESTOSSEN W
    IRD.": :"DIE SENKRECHTE ACHS
    E UND DAS":
232 PRINT "LOCH VERSCHIEBEN SICH
     NACH":"JEDEM STOSS.": :"DAR
    AUS ERGIBT SICH EIN HOHER":
233 PRINT "SCHWIERIGKEITSGRAD,ZU
     EINEM":"TREFFER ZU KOMMEN."
    : : :
234 PRINT "**DRUECKEN SIE EINE T
    ASTE**"
235 CALL KEY(0,K,S)
236 IF S=0 THEN 235
237 IF K>0 THEN 238
238 CALL CLEAR
239 PRINT "**********************
    *******": :
240 PRINT "MAN KANN ALLEINE ODER
     BIS":"ZU 4 TEILNEHMERN DAMI
    T SPIE-":"LEN.": :"JEDER SPI
    ELER ERHAELT 5":"KUGELN":
241 PRINT "EIN TREFFER WIRD MIT
    EINER":"HUEBSCHEN MELODIE AN
    GEZEIGT.": :"NACH ENDE EINES
     SPIELES":
242 PRINT "ZEIGT IHNEN IHR -TI-,
    WER":"DIE MEISTEN TREFFER HA
    T.": : :
245 PRINT "******** VIEL SPASS *
    *******": : :
246 PRINT "**DRUECKEN SIE EINE T
    ASTE**"
247 CALL KEY(0,K,S)
248 IF S=0 THEN 247
249 IF K>0 THEN 280
280 CALL CLEAR
290 PRINT "ANZAHL DER SPIELER!":
     :
300 PRINT "FUER ANZAHL DER SPIEL
    ER":"NUR EINE ZIFFER EINGEBE
    N!": :
310 PRINT ("1,2,3,4")
320 FOR I=1 TO 800
330 NEXT I
340 CALL CLEAR
350 CALL KEY(0,K,S)
360 IF S=0 THEN 350
370 IF CHR$(K)=("1")THEN 410
380 IF CHR$(K)=("2")THEN 410
390 IF CHR$(K)=("3")THEN 410
400 IF CHR$(K)=("4")THEN 410
410 IF CHR$(K)=(" ")THEN 420
420 A$=CHR$(K)
430 B=VAL(A$)
440 GOTO 2620
450 IF B=4 THEN 490
460 IF B=3 THEN 500
470 IF B=2 THEN 510
480 IF B=1 THEN 520
490 INPUT "NAME SP.4>":A4$
500 INPUT "NAME SP.3>":A3$
510 INPUT "NAME SP.2>":A2$
520 INPUT "NAME SP.1>":A1$
530 CALL CLEAR
540 IF B=4 THEN 550 ELSE 590
550 PRINT A4$
560 FOR I=1 TO 250
570 NEXT I
580 CALL CLEAR
590 IF B=3 THEN 600 ELSE 640
600 PRINT A3$
610 FOR I=1 TO 250
620 NEXT I
630 CALL CLEAR
640 IF B=2 THEN 650 ELSE 690
650 PRINT A2$
660 FOR I=1 TO 250
670 NEXT I
680 CALL CLEAR
690 IF B=1 THEN 700 ELSE 730
700 PRINT A1$
710 FOR I=1 TO 250
720 NEXT I
730 CALL CLEAR
740 CALL CHAR(96,"183C7E7E7E7E3C
    18")
750 CALL COLOR(12,2,2)
760 CALL CHAR(127,"0000000000000
    00000")
770 CALL COLOR(12,2,8)
780 CALL VCHAR(4,6,127,19)
790 CALL VCHAR(4,27,127,19)
800 FOR S=52 TO 57
810 CALL VCHAR(S-48,4,S)
```

**Listing zu »Pool-Billard«**

```
820 NEXT S
830 CALL VCHAR(10,4,48)
840 CALL VCHAR(10,3,49)
850 CALL VCHAR(11,4,49)
860 CALL VCHAR(11,3,49)
870 CALL VCHAR(12,4,50)
880 CALL VCHAR(12,3,49)
890 CALL VCHAR(13,4,51)
900 CALL VCHAR(13,3,49)
910 CALL VCHAR(14,4,52)
920 CALL VCHAR(14,3,49)
930 CALL VCHAR(15,4,53)
940 CALL VCHAR(15,3,49)
950 CALL VCHAR(16,4,54)
960 CALL VCHAR(16,3,49)
970 CALL VCHAR(17,4,55)
980 CALL VCHAR(17,3,49)
990 CALL VCHAR(18,4,56)
1000 CALL VCHAR(18,3,49)
1010 CALL VCHAR(19,4,57)
1020 CALL VCHAR(19,3,49)
1030 CALL VCHAR(20,4,48)
1040 CALL VCHAR(20,3,50)
1050 CALL VCHAR(21,4,49)
1060 CALL VCHAR(21,3,50)
1070 FOR S1=1 TO 3
1080 C=INT((22-13+1)*RND)+13
1090 NEXT S1
1120 CALL HCHAR(4,7,127,20)
1130 CALL HCHAR(22,7,127,20)
1140 CALL CHAR(159,"000000000000
     0000")
1150 CALL COLOR(R,2,13)
1160 CALL CHAR(97,"8181818181814
     23C")
1170 CALL HCHAR(22,C,97)
1180 RANDOMIZE
1190 FOR S2=1 TO 3
1200 X=INT((26-17+1)*RND)+17
1210 NEXT S2
1216 FOR L=6 TO 20
1220 CALL HCHAR(L,X,60)
1221 NEXT L
1230 FOR I=1 TO 100
1240 NEXT I
1245 FOR L1=6 TO 20
1250 CALL HCHAR(L1,X,159)
1251 NEXT L1
1260 INPUT "ZEILEN NR.
      -->":U
1270 IF U<4 THEN 1260
1280 IF U>21 THEN 1260
1290 Y=U
1300 XDIR=-1
1310 YDIR=-1
1320 X=X+XDIR
1330 Y=Y+YDIR
1340 IF X<7 THEN 1410
1350 IF X>26 THEN 1410
1360 IF Y<4 THEN 1460
1370 IF Y>20 THEN 1490
1380 CALL HCHAR(Y,X,96)
1390 CALL HCHAR(Y,X,32)
1400 GOTO 1320
1410 XDIR=-XDIR
1420 CALL SOUND(30,500,2)
1430 IF Y<4 THEN 1460
1440 IF Y>20 THEN 1490
1450 GOTO 1320
1460 YDIR=-YDIR
1470 CALL SOUND(30,200,2)
1480 GOTO 1320
1490 IF B=4 THEN 1530
1500 IF B=3 THEN 1790
1510 IF B=2 THEN 2060
1520 IF B=1 THEN 2330
1530 IF X=C+1 THEN 1660
1540 IF X<>C+1 THEN 1550
1550 TONE=220
1560 FOR COUNT=1 TO 3
1570 CALL SOUND(+100,TONE,8)
1580 TONE=TONE+220
1590 NEXT COUNT
1600 FOR I=1 TO 500
1610 NEXT I
1620 F=F+1
1630 F4=F
1640 G=F4+T4
1650 IF G=5 THEN 2570 ELSE 530
1660 TONE=110
1670 FOR COUNT=1 TO 10
1680 CALL SOUND(-500,TONE,3)
1690 TONE=TONE+110
1700 NEXT COUNT
1710 FOR I=1 TO 500
1720 NEXT I
1730 T=T+1
1740 T4=T
1750 G=F4+T4
1760 IF G=5 THEN 2570 ELSE 530
1790 IF X=C+1 THEN 1920
1800 IF X<>C+1 THEN 1810
1810 TONE=220
1820 FOR COUNT=1 TO 3
1830 CALL SOUND(+100,TONE,8)
1840 TONE=TONE+220
1850 NEXT COUNT
1860 FOR I=1 TO 500
1870 NEXT I
1880 F5=F5+1
1890 F3=F5
1900 G1=F3+T3
1910 IF G1=5 THEN 2570 ELSE 580
1920 TONE=110
1930 FOR COUNT=1 TO 10
```

```
1940 CALL SOUND(-500,TONE,3)
1950 TONE=TONE+110
1960 NEXT COUNT
1970 FOR I=1 TO 500
1980 NEXT I
1990 T5=T5+1
2000 T3=T5
2010 G1=F3+T3
2020 IF G1=5 THEN 2570 ELSE 580
2060 IF X=C+1 THEN 2190
2070 IF C<>C+1 THEN 2080
2080 TONE=220
2090 FOR COUNT=1 TO 3
2100 CALL SOUND(+100,TONE,8)
2110 TONE=TONE+220
2120 NEXT COUNT
2130 FOR I=1 TO 500
2140 NEXT I
2150 F6=F6+1
2160 F2=F6
2170 G2=F2+T2
2180 IF G2=5 THEN 2570 ELSE 630
2190 TONE=110
2200 FOR COUNT=1 TO 10
2210 CALL SOUND(-500,TONE,3)
2220 TONE=TONE+110
2230 NEXT COUNT
2240 FOR I=1 TO 500
2250 NEXT I
2260 T6=T6+1
2270 T2=T6
2280 G2=F2+T2
2290 IF G2=5 THEN 2570 ELSE 630
2330 IF X=C+1 THEN 2460
2340 IF X<>C+1 THEN 2350
2350 TONE=220
2360 FOR COUNT=1 TO 3
2370 CALL SOUND(+100,TONE,8)
2380 TONE=TONE+220
2390 NEXT COUNT
2400 FOR I=1 TO 500
2410 NEXT I
2420 F7=F7+1
2430 F1=F7
2440 G3=F1+T1
2450 IF G3>4 THEN 2570 ELSE 680
2460 TONE=110
2470 FOR COUNT=1 TO 10
2480 CALL SOUND(-500,TONE,3)
2490 TONE=TONE+110
2500 NEXT COUNT
2510 FOR I=1 TO 500
2520 NEXT I
2530 T7=T7+1
2540 T1=T7
2550 G3=F1+T1
2560 IF G3>4 THEN 2570 ELSE 680
2570 Q=B+1
2580 Q=Q-1
2590 IF Q=1 THEN 2640 ELSE 2600
2600 B=B-1
2610 GOTO 530
2620 LET Z=B
2630 GOTO 450
2640 CALL CLEAR
2650 IF Z=4 THEN 2690 ELSE 2660
2660 IF Z=3 THEN 2710 ELSE 2670
2670 IF Z=2 THEN 2730 ELSE 2680
2680 IF Z=1 THEN 2750
2690 PRINT A4$: :
2700 PRINT "TREFFER:";T4;"FEHL:"
     ;F4: : :
2710 PRINT A3$: :
2720 PRINT "TREFFER:";T3;"FEHL:"
     ;F3: : :
2730 PRINT A2$: :
2740 PRINT "TREFFER:";T2;"FEHL:"
     ;F2: : :
2750 PRINT A1$: :
2760 PRINT "TREFFER:";T1;"FEHL:"
     ;F1: : :
2770 FOR I=1 TO 2000
2780 NEXT I
2790 CALL CLEAR
2800 PRINT "NOCH EIN SPIEL(J/N)?
     "
2810 GOSUB 2890
2820 CALL KEY(0,K,S)
2830 IF S=0 THEN 2820
2840 IF K=74 THEN 2860
2850 IF K=78 THEN 2880
2860 CALL CLEAR
2870 GOTO 300
2880 END
2890 G3=0
2900 T1=0
2910 F1=0
2920 F7=0
2930 T7=0
2940 G2=0
2950 T2=0
2960 F2=0
2970 T6=0
2980 F6=0
2990 G1=0
3000 T3=0
3010 F3=0
3020 T5=0
3030 F5=0
3040 G=0
3050 T4=0
3060 F4=0
3070 T=0
3080 F=0
3090 RETURN
```

**Listing zu »Pool Billard« (Schluß)**

bestimmt werden. Die Kugel bewegt sich zunächst in einem 45-Grad-Winkel nach oben. Der Auf- und Abprallwinkel beträgt dann stets 90 Grad. Treffer und Fehlschüsse werden während des Spiels akustisch untermalt.

Jeder der bis zu maximal vier Spieler erhält zu Spielbeginn fünf Kugeln, die erst verschossen werden müssen, bevor der nächste Mitspieler an der Reihe ist.

Nun noch eine kurze Anregung, wie man das Spiel noch spannender und unterhaltsamer gestalten kann: Jeder Teilnehmer zahlt einen kleinen Betrag in eine Kasse, die dann an den Gewinner ausgezahlt wird. Haben zufällig zwei Spieler die gleiche Anzahl von Treffern, wird ein Stechen durchgeführt.

(L. Hoffmann)

## Sprachausgabe für 64

In Heft 1/84 wurde nach einer Sprachausgabe für den Commodore 64 gefragt. Ich besitze ein Programm, das den Befehlssatz des Commodore 64 um den Befehl »Say« erweitert. Gibt man diesen Befehl im Direktmodus oder im Programm mit Wörtern dahinter ein, so »spricht« der 64 die Wörter aus. Ich gebe das Programm an Interessenten mit Demos und einer Kurzbeschreibung gegen Einsendung von 20 Mark und einer Kassette beziehungsweise Diskette ab. Meine Adresse: Lutterdamm 130, 4540 Bramsche 1 Thomas Tai

## Roboter-Club

Ich bin Physikstudentin und begeisterter Computer-Freak, der soeben damit begonnen hat, einen Roboter zu basteln. Ich halte die Mechanik eines Roboters für wesentlich komplizierter als die Computer-Steuerung. Hat man sich durch die Mechanik durchkonstruiert, kommt die nächste nicht zu verachtende Schwierigkeit, die Materialbeschaffung. Ich würde mich sehr freuen über einen Roboter-Club günstige Bezugsquellen oder gar Baukästen zu bekommen und an einem Informationsaustausch teilzunehmen.

Katja Balog

Ich bin an der Roboter-Technik interessiert und würde mich freuen, Informationen über Bausätze, Baupläne und Bezugsquellen zu bekommen.

Helmut Mizgaiski

Mitglieder des Computer-User-Clubs Salzburg »COMUSCLUS« (Postfach 128, 5033 Salzburg) befassen sich derzeit mit dem Aufbau eines Roboters.

Die Firma rhv softwaretechnik gmbh (Georg-Glock-Str. 3, 4000 Düsseldorf 30) hat ein »Robot-Democenter« eingerichtet. Dort werden 3 Tage-Intensiv-Seminare durchgeführt, die Fachwissen über Einsatzmöglichkeiten von Industrie-Robotern vermitteln.

## AND und OR beim 99/4

Ich habe festgestellt, daß die Funktionen »AND« und »OR« zwar nicht im Befehlssatz des TI-Basic (oder des Extended-Basic) enthalten sind, man kann die gleiche Wirkung jedoch mit Rechenoperationen erzielen. Diese Methode ist in den meisten Büchern nicht bekannt, dort wird darauf verwiesen, mehrere IF-Zeilen zu benutzen. Diese Methode ist aber etwas kompliziert und benötigt auch mehr Speicherplatz (da mehrere Zeilen belegt werden und IF und THEN immer neu geschrieben werden müssen). Deshalb wäre meine Methode vorzuziehen.
Beispiele:
100 IF A=15 AND B=8 OR C=3 THEN 200 (Microsoft)
Umschreibversuche:
1. nach Literatur:
100 IF A=15 THEN 101 ELSE 105
101 IF B=8 THEN 200
105 IF C=3 THEN 200
2. meine Lösung:
100 IF (A=15)★(B=8)+(C=3) THEN 200
weiteres Beispiel:
100 IF B=C AND A=D THEN 500 (Microsoft)
100 IF (B=C)★(A=D) THEN 200
»★« entspricht also der UND-Funktion, »+« entspricht der ODER-Funktion. Die Verzweigungen der anderen Rechenoperatoren sind in der folgenden Tabelle zusammengestellt:

x: Verzweigung; -: keine Verzweigung; w: Warnung; r: richtig; f: falsch; 1.: 1. Aussage; 2.: 2. Aussage

| Rechenoperator | 1.r und 2.r | 1.r, 2.f | 1.f, 2.r | 1.f, 2.f |
|---|---|---|---|---|
| + | x | x | x | - |
| - | - | x | x | - |
| ★ | x | - | - | - |
| / | x | wx | - | wx |
| ^ | x | x | wx | x |
| = | x | - | - | x |
| < | - | x | - | - |
| > | - | - | x | - |
| <= | x | x | - | x |
| >= | x | - | x | x |

Klaus Schneider

## EXBASIC erläutert

In Heft 12/83 fragte Michael Kuhne unter anderem nach Erläuterungen der neuen Basic-Befehle der Basic-Erweiterung EXBASIC LEVEL II für VC 20. Das Handbuch (120 Seiten, gebunden) kann zum Preis von 50 Mark bei jedem Commodore-Händler, im Buchhandel oder beim Distributor Interface Age (Vohburgerstr. 1, 8000 München 21, Tel. 089/5806702) bezogen werden (Computertyp »VC 20« mit angeben). Jeder rechtmäßig erworbenen Programmversion des EXBASIC LEVEL II liegt ein Handbuch bereits bei.

Andreas Dripke

## Mischen von Programmen

In Heft 12/83 fragten mehrere Leser nach Befehlen zum Mischen von Programmen im Basic-Speicher. Wir möchten darauf aufmerksam machen, daß ein solcher Befehl »Merge« nicht nur unter Simons Basic verfügbar ist, sondern auch unter EXBASIC LEVEL II. Der EXBASIC-Befehl erlaubt sogar nicht nur das Aneinanderhängen von Programmen, sondern darüber hinaus das ineinander Vermischen einzelner Programmzeilen von Kassette oder Diskette.

Andreas Dripke

## Colour-Genie-Tips

Wir möchten hier versuchen, so viele Leseranfragen wie möglich zu beantworten. Auf Patric Lagnys Frage nach der Maximalerweiterung des Colour-Genie können wir ihm die Mitteilung machen, daß es im Sommer 1984 möglich sein soll, das Colour-Genie auf 96 KByte Speicherkapazität aufzurüsten.

Und nun zu den Fragen von Peter Mees: Es ist möglich, Umlaute und das »ß« auf Tastatur zu legen. Eine Basic-Lösung und eine Maschinensprachelösung werden wir in einer unserer nächsten Ausgaben von »ZIETLOW-GENIE« veröffentlichen. »ZIETLOW-GENIE« ist ein Versuch, etwas gegen die auch von Frank Burkatzky in Happy Computer 2/84 angesprochene Literaturflaute zu tun. Es ist eine Zeitschrift, die sich ausschließlich mit dem Colour-Genie beschäftigt und von uns publiziert wird. Ebenso werden wir ein von Robert Zimmermann in Happy Computer 1/84 angesprochenes Programm bringen, das Text im FGR-Modus plottet. Informationen und eine Softwareliste können bei uns gegen Zusendung von 1 Mark in Briefmarken bezogen werden. Unsere Anschrift: SWH Zietlow, Wilhelm-Kraft-Str. 14, 4322 Sprockhövel 2, Tel. 02339/3442

H. Zietlow

## Vorsicht mit beidseitiger Nutzung

Mit Ihrem Artikel über die Verdoppelung der Speisekapazität durch beidseitige Nutzung von Floppydisks, die vom Hersteller nur für einseitige Benutzung vorgesehen sind, haben Sie dem Billigmarkt eine gute Anregung gegeben, aber auch zugleich einen schlechten Dienst getan. Nach »Check Disk« auf der 1541-Floppy von Commodore (welches wegen 1,5 h Dauer eine Tortour ist), stellte ich leider fest, daß mit Ausnahme eines Teils der Nashua-Disketten und Scotch-Disketten alle geprüften Disketten Blockfehler aufweisen. Bei der Bequemlichkeit der Computerfreunde ist es sicher, daß ein Großteil einfach im guten Glauben die Rückseite der Disketten nun mitbenutzen wird. Ergebnis: Lauter defekte Programme in der eigenen Sammlung und auf dem Tauschmarkt. Da sollte man schon auf DS-Disketten (Multilife) zurückgreifen (oder die Disketten prüfen ...). Nicht umsonst kosten Profi-DS-Disketten das Doppelte — sie haben auch kein Übersprechen von der einen Seite auf die andere und sind kratzfest!

M. Buchholz

## VC 20/64: Poke-Befehle umschreiben

In Heft 12/83 wurde gefragt, wie man VC 20-Poke-Befehle für den Commodore 64 umschreiben könne. In dem von uns verlegten Buch »Das Interface Age Sytemhandbuch zum Commodore 64 und VC 20« erläutert alle Betriebssystemunterschiede. Es enthält auch einen Vergleich der ROM-Bereiche sowie eine Liste, aus der die — wenigen — Unterschiede der Poke-Befehle im RAM-Bereich ersichtlich sind. Das jetzt in zweiter Auflage erschienene Buch ist für 74 Mark bei den Commodore-Händlern, im Buchhandel oder beim Distributor Interface Age (Vohburger Str. 1, 8000 München 21) erhältlich. Andreas Dripke

Bildschirm gebracht und die zugehörigen Funktionen nicht ausgeführt, sondern gespeichert, was man beim 8032 durch Drücken der ESC-Taste abschalten konnte. Die kleine Maschinenroutine bewirkt das gleiche beim Drücken der C-(=Commodore-)Taste des Commodore 64.

Das Programm belegt einen Teil des Kassettenpuffers. Vor Benutzung des Kassettenrecorders sollte deshalb immer mit SYS 839 auf Normalzustand geschaltet werden, da sonst unvorhersehbare Folgen eintreten können (das Programm hängt direkt am Interrupt). Falls die Basic-Erweiterung »Simons Basic« verwendet wird, funktioniert obige Routine nicht, da in Simons Basic ebenfalls der Interrupt verstellt wird.

(Lennart Kaschella)

```
828 LDA =82      169   82          ; Veränderung
830 STA 788      141   20     3    ; des IRQ-Vektors
833 LDA =3       169    3          ;
835 STA 789      141   21     3    ;
838 RTS           96                ;
839 LDA =49      169   49          ; Wiederherstellung
841 STA 788      141   20     3    ; des IRQ-Vektors
844 LDA =234     169   234         ;
846 STA 789      141   21     3    ;
849 RTS           96                ;
850 LDA 145      165   145         ; C= —Taste gedrückt?
852 CMP =223     201   223         ;
854 BNE862       208     6         ; nein, zurück
856 LDA =0       169     0         ;
858 STA          133   212         ; '"'-Flag löschen
860 STA 216      133   216         ; INSERTs löschen
862 JMP 59953     76    49   234   ; zum Interrupt
```

**Basic-Lader der Escape-Routine**

```
10  DATA 169,82,141,20,3,169,3,141,21,3,96,169,49,141,20,3,169,234,141,21,3,96
20  DATA 165,145,201,223,208,6,169,0,133,212,133,216,76,49,234,
30  FOR I = 828 TO 864: READA:S=S+A:POKEI,A:NEXT
40  IF S# > <4064 THEN PRINT "FEHLER IN DATA-BLOCK":LIST
50  PRINT"EINSCHALTEN: SYS828"
60  PRINT"AUSSCHALTEN: SYS839"NEW
```

**Assemblerlisting der Escape-Routine**

## Viele Leser haben es gemerkt: Unser HELP-Programm in der April-Ausgabe von Happy-Computer war ein April-Scherz. Es verleiht dem TI 99/4A in der Tat eine sehr eigenwillige Intelligenz.

```
270 CALL ENCODE(SCODE,TEXT1$,
    UNTEXT$)
280 PRINT "GEHEIMTEXT: "&UNTE
    XT$: :
290 INPUT "PASSWORT (ENTSCHLU
    ESSELN):  ":SDECODE :: PR
    INT
300 CALL DECODE(SDECODE,UNTEX
    T$,TEXT2$)
310 PRINT "KLARTEXT: "&TEXT2$
    : :
320 IF TEXT1$()TEXT2$ THEN PR
    INT "FALSCH! NEUES" :: GO
    TO 290
330 PRINT "PASSWORT RICHTIG!"
    : : : : : : :
340 GOTO 250
350 !----------------------
360 ! SUBPROGRAMM-MODULE
370 !----------------------
380 SUB ENCODE(SCODE,TEXT$,UN
    TEXT$)!          ********
    **********************
390 RANDOMIZE SCODE :: UNTEXT
    $="" :: V=ABS(SCODE/10-IN
    T(SCODE/10))*10
400 FOR LOOP=1 TO LEN(TEXT$)
410 X=INT(RND*95-V):: A=ASC(S
    EG$(TEXT$,LOOP,1))
420 IF (A+X)(128 THEN B$=CHR$
    (A+X)ELSE B$=CHR$(A+X-95)
430 UNTEXT$=UNTEXT$&B$
440 NEXT LOOP
450 SUBEND
460 SUB DECODE(SDECODE,UNTEXT
    $,TEXT$)!        ********
    **********************

470 RANDOMIZE SDECODE :: TEXT
    $="" :: V=ABS(SDECODE/10-
    INT(SDECODE/10))*10
480 FOR LOOP=1 TO LEN(UNTEXT$
    )
490 X=INT(RND*95-V):: A=ASC(S
    EG$(UNTEXT$,LOOP,1))
500 IF (A-X))31 THEN B$=CHR$(
    A-X)ELSE B$=CHR$(A-X+95)
510 TEXT$=TEXT$&B$
520 NEXT LOOP
530 SUBEND
```

auch eine Variable sein darf, kann durch eine einfache Verschlüsselungsroutine (siehe Listing) eine Scramble-Funktion mit Passwort-Eingabe realisiert werden (als Passwort dient eine beliebige Zahl).

Übrigens: alle PEEK- und LOAD-Befehle sind reiner Bluff. Der Wert F in den Zeilen 490 bis 530 entspricht jeweils dem einfachen Summanden. In Zeile 240 wird ein Wert aus der Adresse 2454 gePEEKt und auch gleich wieder gePOKEt (3*818=2454!) Die Variablen D, Q und XX sind wie N nur Scheinvariable mit dem Wert 0. Die Ausgabetexte in der DATA-Zeile 650 sind potemkinsche Dörfer. Sie werden zwar zuerst in die Ausgabevariablen HILF$( ) geladen, dann aber in Zeile 660 durch die unverschämten Texte aus den Variablen P$( ) ersetzt.

Die Hilfsroutine am Programmende kann immer dann verwendet werden, wenn längere Datazeilen mit Maschinencodes zu programmieren sind. Das Programm bricht dann bei fehlerhaften DATA-Werten ab, ohne daß der Computer sich aufhängt. In Zeile 350 sind zwei Besonderheiten versteckt, mit denen die INPUT-Anzeige stumm und fragezeichenlos gemacht wird. Die Ausgabe des Fragezeichens verhindert der Prompt mit dem Leerstring (die zwei unmittelbar aufeinanderfolgenden Anführungszeichen nach »INPUT«). Eine Stummschaltung des Anforderungspieps erreicht man durch einen vorangestellten SOUND-Befehl, dessen Lautstärke mit »30« auf Minimum gestellt wird und dessen Dauer länger als der INPUT-Pieps sein muß.

Alle Hinweise im Text auf die »gefährlichen« Pokes und Beeinflussungen des ROM-Basics durch falsche Werte waren natürlich Humbug. Auch »Veränderungen« durch manuelle Programmstarts sollten nur ein vorzeitiges Aufdecken des wahren Charakters verhindern. Und Systemadressen wurden schon gar nicht verwendet, weder unbekannte noch bekannte.

(I. Maier)

**Die einfache Scramble-Routine zum Ver- und Entschlüsseln von Texten erlaubt als Pass-»Wort« eine Zahl zwischen −9999999999999 bis 99999999999999, deren letzte Stelle noch unterschieden wird! Natürlich kann auch ein echtes Wort benützt werden, wenn dieses nach Eingabe (zum Beispiel über den ASCII-Wert der Buchstaben) in einen numerischen Wert umgewandelt wird. Die eigentlichen Scramble-Module sind in den Zeilen 380 bis 530 enthalten. Sie können an das Hauptprogramm angehängt werden.**

# Trickreiche Lade- und Save-Hilfe

Es gibt Situationen, in denen es sinnvoll ist, Daten nicht als sequentielles Datenfile, sondern als Programmfile zu speichern und zu laden.

Nach der manchmal sehr zeitraubenden Berechnung und Darstellung von mathematischen Funktionen oder sonstigen Grafiken besteht oft der Wunsch, diese auf Kassette oder Diskette zu speichern. Die hier aufgeführten Routinen übernehmen diese Aufgabe bei optimaler Speicherausnutzung und damit hoher Übertragungsgeschwindigkeit.

Für das Abspeichern und Laden bietet der VC 20 zwei Möglichkeiten:

**1) Eröffnen eines sequentiellen Datenfiles.**

Hierbei müßte eine Programmschleife mit PEEK beziehungsweise POKE und PRINT #n gebildet werden. Dies ist umständlich und wirkt sich negativ auf die Geschwindigkeit aus. Der größere Nachteil ist aber

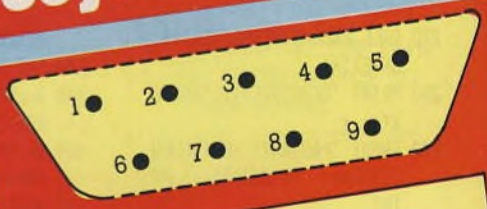
```
10 rem save-hilfe
20 print"file-name":input a$
30 poke187,0:poke188,2:rem adresse file-name
40 poke183,len(a$)
50 poke185,1:rem sekundaeradresse
60 poke186,8:rem floppy
70 poke193,0:poke194,16:rem beginn graphic-speicher
80 poke174,128:poke175,28:rem ende+1
90 sys63106
ready.
```

**Speicher-Routine**

# Belegung der Joystick-Buchse

Die Joysticks von Texas Instruments zum TI 99/4A waren schon früher nicht gerade billig, jetzt gibt es sie gar nicht mehr. Da bleibt eigentlich nur ein Selbstbau oder die Anpassung anderer Fabrikate.

Der 9polige Stecker ist im Elektronikhandel leicht zu beschaffen. Lediglich die Anschlußbelegung ist vielen Anwendern ein Rätsel. Selbst die technischen Unterlagen von Texas Instruments weisen in dieser Beziehung Fehler auf. Deshalb zeigen wir hier, wie die Kon-




| | |
|---|---|
| 7 | Eingang Joystick 1 |
| 2 | Eingang Joystick 2 |
| 9 | X-Koordinate = +4 |
| 5 | X-Koordinate = —4 |
| 3 | Y-Koordinate = +4 |
| 8 | Y-Koordinate = —4 |
| 4 | K:ASCII-Code Feuerknopf = 18 |

**Beschaltung der Joystickbuchse beim TI 99/4A (Aufsicht auf die Buchse am Computer)**

ein, gegenüber dem ursprünglich benötigten Speicherplatz, erheblich umfangreicherer Speicherplatzbedarf. Die Zahlen (0...255) werden als String mit Vorzeichen und »Cursor right« abgelegt, für eine einstellige Zahl werden daher schon drei Bytes benötigt. Wesentliche Vorteile bietet dagegen folgende Methode:

**2) Ablegen als Programm-file**

Der Inhalt des Grafikspeichers ($1000...$1C80) wird wie eine Maschinenroutine behandelt und als solche, wie schon mehrfach beschrieben, abgelegt. Die Verwendung des Befehls »load« für das Laden bringt Schwierigkeiten, da am Ende dieser Routine der Pointer für den Variablenstart auf das »scheinbare« Programmende gesetzt wird und ein Basic-Warmstart erfolgt. Daher ist es hier besser, durch einen Sprung in die Maschinenebene einen Teil der Routine zu umgehen.

Als Zwischenspeicher für den File-Namen wird der Basic-Eingabepuffer ab $0200 verwendet. Somit lassen sich die beiden Routinen leicht in jedes Basicprogramm einbinden. Die Lade-Routine (SYS 62795) darf natürlich erst nach dem Befehl »graphic n« ausgeführt werden; zu berücksichtigen ist dann noch das Füllen des Farbspeichers.

(Hermann Weißenberger)

```
10 rem lade-hilfe
20 print"file-name":input a$
30 poke187,0:poke188,2
40 poke183,len(a$)
50 poke185,1:rem #sa umgehen des verschiebeladers
60 poke186,8:rem floppy
70 poke10,0:poke147,0:rem flag load/verify
80 sys62795
ready.
```

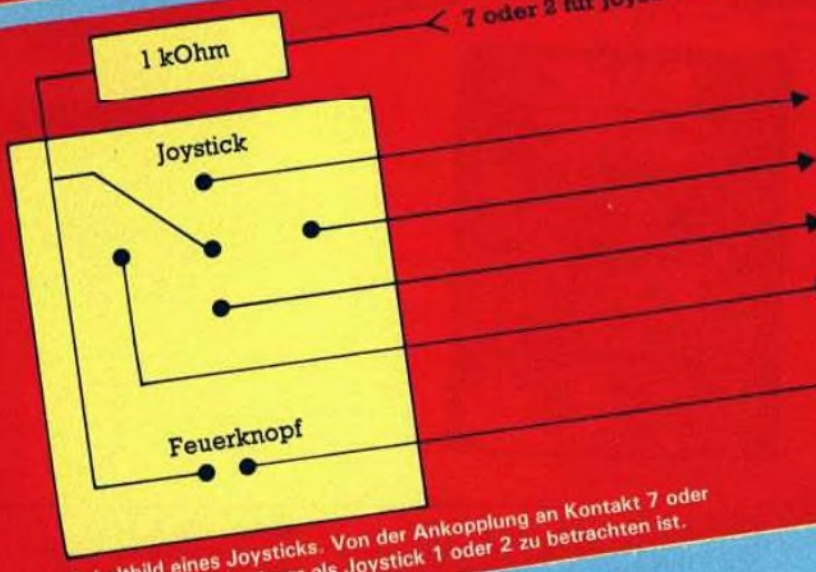
**Lade-Routine**

# beim TI 99/4A

takte angeschlossen werden müssen. Mit folgender Testroutine kann die Beschaltung nach dem Zusammenlöten überprüft werden:

```
10 CALL CLEAR
20 CALL JOYST(1,X1,Y1)
30 CALL KEY(1,K1,S)
40 CALL JOYST(2,X2,Y2)
50 CALL KEY(2,K2,S)
60 PRINT "1=X";X1;",Y";Y1;",K";K1;"*2=X";X2;",Y";Y2;",K",K2
70 GOTO 20
```

In die Leitung zu den Joysticks sollte vorsichtshalber je ein Schutzwiderstand eingeschleift werden. Sein Wert kann zwischen 330 Ohm und 2,2 kOhm liegen. Ein optimaler Wert ist 1 kOhm. (lg)




Schaltbild eines Joysticks. Von der Ankopplung an Kontakt 7 oder 2 hängt es ab, ob dieser als Joystick 1 oder 2 zu betrachten ist.

# TI 99/4A

## -früher Frust statt lange Lust?

**Noch Ende vergangenen Jahres versprach Texas Instruments den Käufern seines Heimcomputers TI 99/4A neue Software und Hardware. Kurz vor Weihnachten wurde dann die Einstellung der Produktion beschlossen.**

Die restlichen Lagerbestände wurden spottbillig verramscht. Der Effekt: Viele kauften die preiswerte Konsole, ohne als Einsteiger absehen zu können, daß sie ohne zusätzliche Peripherie bald schon — und viel zu früh — an die Grenzen des Computers stoßen würden.

In den USA gibt es für den TI-Heimcomputer-Markt schon seit langem unzählige Fremdhersteller mit einem Riesenangebot an Soft- und Hardware. Die Entscheidung des US-Mutterkonzerns hat dort für den Anwender also kaum Konsequenzen. Im Freisinger Zweigwerk löste die Einstellung der Produktion aber sicher keine guten Gefühle aus. Durch die Struktur des deutschen Markts bedingt, steht der hiesige Anwender nämlich viel hilfloser da, als sein amerikanischer Leidensgenosse.

### Nachschub: In den USA kein Problem

Noch sind zwar nicht alle Lager mit TI 99/4A-Zubehör im Handel geleert, noch führt der eine oder ander Händler das eine oder andere Originalzubehör. Auch in den Kleinanzeigenteilen einschlägiger Computerzeitschriften werden Teile angeboten. Aber auf diesen Wegen genau jenes Zubehör zu finden, das man für den eigenen Computer sucht, ist ein Spiel mit dem Zufall und glückliche Zufälle werden von Tag zu Tag seltener. Deshalb versuchen wir nicht, zu zeigen, wo vielleicht noch ein bestimmtes Originalzubehör zu finden ist, sondern, wo neue Quellen entstehen, die unabhängig von Restbeständen Zubehör anbieten. Bei diesen Quellen wird nämlich einzig die Nachfrage aus den Reihen der TI-Besitzer darüber entscheiden, wie lange es ein bestimmtes Teil geben wird.

Für den Anwender, so zeigten unsere Recherchen, herrscht zur Zeit eine besonders ungünstige Situation. Zum einen gibt es kaum noch Originalteile, zum anderen können aber die meisten Fremdhersteller noch nicht liefern, teils weil kleine Firmen für umfangreichere Neuentwicklungen nur eine begrenzte Personalkapazität besitzen (nicht selten ist der Inhaber die ganze Firma), teils weil entsprechende Lizenzverhandlungen mit Texas Instruments noch nicht abgeschlossen sind. Deshalb gibt es zum Beispiel auch noch keine neuen Extended-Basic-Module am Markt. Für die Software in den Modulen besitzt Texas Instruments die Urheberrechte. Die langen Verhandlungen haben für den Anwender zwei Seiten. Einerseits muß er deshalb vielleicht noch ein viertel Jahr auf sein Modul warten, andererseits will Texas Instruments die Lizenzen nur an eingeführte Unternehmen vergeben. Der Vorteil für den Anwender: eine gewisse Sicherheit, daß der ausgewählte Hersteller nicht schon nach wenigen Monaten vom Markt verschwindet, samt Produkt, Service und Garantie.

Wir können daher nur wenige Produkte vorstellen, die bereits jetzt erhältlich sind. Die meiste neue Peripherie wird erst in einigen Wochen oder Monaten auf den Markt kommen.

Etwas grotesk ist immerhin, daß die Anwender des TI 99/4A in einem halben Jahr mit großer Wahrscheinlichkeit aus einer vielfältigeren Produktpalette wählen können als zu »Lebzeiten« der TI-Heimcomputer-Produktion. Das funktioniert allerdings nur bei entsprechender Nachfrage und Gewinnspanne für die Hersteller. Te-

**Diskettenstationen (erhältlich):**

| | |
|---|---|
| BASF-Laufwerke, doppelseitig, 180 KByte (nur für Controller 2 in Expansion Box) | 898 Mark |
| ohne Gehäuse | 1098 Mark |
| mit Gehäuse | 1798 Mark |
| 2 Laufwerke in einem Gehäuse | |

**Software (erhältlich):**

| | |
|---|---|
| »Damast« (Datenmanagement-System für Konsole 32-KByte-RAM-Erweiterung, Diskettenstation, Extended Basic) | ab 218 Mark |
| »Expert« (Texteditor für Konsole) | |
| Kassette | 89 Mark |
| Diskette | 99 Mark |

**Speichererweiterung (wird erwartet):**

| | |
|---|---|
| 64 KByte-RAM-Erweiterung mit Centronics-Schnittstelle | Preis ungewiß |
| außerdem Produkte der Firmen Klein und Atronik | |

Info: Das Computer-Hüsli, Postfach 860808, 8000 München 82, Telefon (089) 4300353

**Expansion-Box (wird erwartet):**

| | |
|---|---|
| Peripheriebox mit fest montierten Karten 32-KByte-RAM-Erweiterung, 2 RS232- und 1 Centronics-Schnittstelle, 90-KByte-Diskettenlaufwerk mit Controller soll kommen | ca. 2500 Mark |

**Extended-Basic-Modul (in Planung):**

| | |
|---|---|
| Auch ein Extended-Basic-Modul ist geplant | Preis ungewiß |

Info: Schuh & Co., Goethestraße 17, 8034 Germering, Telefon (089)843064

# Hilfe für TI-Benutzer

**Kabel (erhältlich):**

| | |
|---|---|
| Adapter für zwei Joysticks (Atari u. ä.) | 22 Mark |
| Kassettenrecorderanschluß | 39 Mark |

**Software (Archivprogramme erhältlich):**

| | |
|---|---|
| »Darts«, »Zentoria«, »Monko«, »Digger« (Spiele) | je 39 Mark |
| »City Quiz« (Lernprogramm) | je 49 Mark |
| »Schallplatten Archiv«, »Bücherarchiv« | je 69 Mark |
| »Vokabel Trainer« (Lernprogramm) | je 98 Mark |
| »Adressen Archiv« und »Video Archiv« | |
| »Data-Bank« und »Lager Kartei« (Datenverwaltung) | |

Info: Dynamics, Postfach 112005, 2000 Hamburg 1

**Speichererweiterung (erhältlich):**

| | |
|---|---|
| Erweiterungskarte in CMOS-Technik für den Einbau in die Konsole (Achtung Löterfahrung nötig) | 250 Mark |
| ohne RAM-Chips | 342,50 Mark |
| mit 16 KByte RAM | 450 Mark |
| mit 32 KByte RAM | |

**Schnittstellen (erhältlich):**

| | |
|---|---|
| Centronics-Schnittstelle (Anschluß ist kompatibel mit dem PIO-Ausgang von TI) | 250 Mark |
| Karte für Peripheriebox | 350 Mark |
| Schnittstellen-Box mit eigenem Netzteil | |

**Speichererweiterung und Diskettencontroller (in Planung):**
In Planung sind eine 32-KByte-Erweiterung für die Peripheriebox und ein Diskettencontroller (doppelseitig und doppelte Dichte) als stand-alone-Gerät.

Info: Franz Elektronik, Seestraße 159, 2083 Halstenbek, Telefon (04101) 46778.
Die Schnittstellen werden auch angeboten von: Radix Bürotechnik, Bornstraße 4, 2000 Hamburg 13, Telefon (040) 441695-96

## Nach Redaktionsschluß eingegangen:

Deutsches Editor/Assembler-Handbuch, Wycobe-Forth (Compiler auf Diskette für Konsole und Diskettenlaufwerk, mit deutschem Handbuch) 120 Mark

Zeta-de-luxe-Joystick (Atari-ähnlich, direkt anschließbar) pro Stück 49 Mark
weiteres Zubehör, Kabel, Adapter u. ä.
Wickanderware, Müllerstr. 59a, 1000 Berlin 65, Telefon (030) 4512788

Printer/Plotter (Papierbreite 14 mm, Tintenkugelschreiber-Prinzip, Centronics-Schnittstelle) ca. 450 Mark
Diskettenstation mit Controller in Planung (voraussichtlich Ende April)
Mechatronik, Dresdner Str. 21, 7032 Sindelfingen, Telefon (07031) 875042

xas Instruments beabsichtigt hier im Nachhinein noch etwas Gutes für seine ehemaligen Kunden zu tun, indem es die Lizenzvergabe nach eigenen Angaben großzügig handhabt.

Unter den bisherigen Anbietern des TI 99/4A im Handel haben sich zwei Lager gebildet. Die einen reagieren knallhart und geben unverblümt zu, daß für sie mit dem TI 99/4A auch der Handel mit der Peripherie gestorben ist. »Wissen Sie, die Leute gehen doch von Geschäft zu Geschäft und fragen nach den Sachen. Wenn wir die dann wirklich hätten, gäb's kaum jemanden, der sie tatsächlich kaufen würde«, so der Sprecher eines Kaufhauskonzerns.

Da lobt man sich die Ausnahmen: Zwei Kaufhauskonzerne scheinen Verpflichtung gegenüber ihren Kunden zu empfinden. Beim Kaufhof kaufte man alle Restbestände an Zubehör auf, deren man habhaft werden konnte und orderte darüber hinaus neue Peripherie von Fremdherstellern. Auf Befragen erklärte ein Vertreter des Unternehmens, daß man beabsichtige, auch in Zukunft Peripherie und Kleinzubehör für den TI 99/4A anzubieten. Zur Zeit gäbe es allerdings gewisse Engpässe, zum Beispiel beim Extended-Basic-Modul, das momentan nur als Toolkit (Modul, Handbuch und Sprachmodul) lieferbar sei. Es wären aber in den meisten Filialen originale Software, Kassettenrecorder-Kabel und Joysticks, sowie eine Centronics-Druckerschnittstelle eines Fremdherstellers vorhanden. Auch bei Karstadt will man sich um Produkte von solchen Fremdherstellern bemühen. Gewiß: Diese Haltung ist nicht selbstlos. Sie entspricht eher klugem Kaufmannsdenken klassischer Art. Die Kunden werden eine solche »Fürsorge« mit Kundentreue auch bei anderen Produkten belohnen und damit indirekt das Risiko zu geringen Absatzes aufwiegen. Da geht die Rechnung dann sogar nach »Heller und Pfennig« auf und nützt obendrein dem Verbraucher.

Das gleiche Bild bei der Software: Für Ariola gilt der Markt derzeit als faktisch tot, Software für den TI 99/4A wird nicht mehr produziert.

## Neue Software bereits in Sicht

Bei Teldec wirkt man etwas unentschlossen, da man zur Zeit ein Spiele-Modul mit dem populären »Miner 2049« für 149 Mark anbietet und viele Anfragen nach Software bekommt — der Absatz aber zu

**Anschlußkabel (erhältlich):**

| | |
|---|---|
| für einen Kassettenrecorder (Klinken- oder 5poliger DIN-Stecker) | 29 Mark |
| wie oben aber für zwei Recorder | 45 Mark |
| für Walkman (zusätzlich Adapter) | 6,50 Mark |
| für zwei Joysticks (Atari, Commodore u.ä.) | 32 Mark |
| Verlängerungskabel 2,50 m | 12 Mark |

Info: Axel Rinke, Bauerstr. 2, 8000 München 40, Telefon (089)271 4902 (oder (089)3009402 ab 18 Uhr).

wünschen übrig läßt. Immerhin soll im Mai ein brandneues Spiel als weiteres Modul angeboten werden: »Espial«, ebenfalls für 149 Mark. Bei Atari sollen ab Ende April sogar ganze fünf neue Module erscheinen (»Jungle Hunt«, »Joust«, »Moon Patrol« und »Mrs. Pacman« für jeweils 99 Mark, sowie »Pole Position« für 119 Mark).

## Große Nachfrage sichert Angebot

Eines wäre für die Anwendergemeinde sicher verhängnisvoll: zu resignieren. Nur so lange wie die zahlreichen Benutzer des TI 99/4A ihre Wünsche nach neuem Zubehör laustark formulieren, wird für den Handel ein Anreiz bestehen, nach entsprechenden Produkten zu suchen und sie in sein Angebot aufzunehmen. Von Texas Instruments ist zu hören, daß sich immerhin ein knappes Dutzend ernstzunehmender Bewerber um die Lizenzen bemühen. Unter anderem könnten auf diese Weise in einigen Monaten sowohl das Extended-Basic-Modul wieder auf den Markt kommen, als auch das Minimem und der Editor-Assembler. Die Namen der Bewerber waren aus Wettbewerbsgründen leider noch nicht zu erfahren.

Wer nicht solange warten will, dem bleibt noch der Gebrauchtmarkt. Das Angebot an gebrauchter Peripherie ist zwar groß, aber kleiner als die Nachfrage. Besonders zwei Zusätze sind besonders begehrt: das Extended-Basic-Modul, für das gegenwärtig mehr als der Neupreis gezahlt wird (bis zu 450 Mark), und die 32-KByte-Erweiterung. Wir konnten bei einem einzigen Inserat 16 Anfragen nach dem Extended-Basic-Modul und 8 Anfragen nach der RAM-Erweiterung registrieren. Dabei birgt so ein Kauf eines gebrauchten Geräts für den Käufer erhebliche Risiken. Heimcomputer sind nicht besonders robust und die Instandsetzung ist sehr teuer. Texas Instruments verlangt zum Beispiel für eine Reparatur der Konsole pauschal 250 Mark, fast soviel, wie diese zuletzt neu kostete. Für Bastler sind aber unter Umständen sogar defekte Geräte interessant, da sich diese als Ersatzteillager für ein schon vorhandenes Gerät ausschlachten lassen. Wenig gefragt ist exotischere Pheripherie, wie zum Beispiel der Assembler oder Diskettenlaufwerke.

Texas Instruments selbst garantiert die Einhaltung der sechsmonatigen Garantiezeit und darüber hinaus Service- und Reparaturleistungen für einige Jahre. Die meisten Geräteteile stammen ohnehin aus laufenden Fertigungen und werden daher nach wie vor produziert.

**Speichererweiterung (erhältlich):**
32-KByte-RAM-Erweiterung mit Gehäuse
mit CMOS-RAMs, können durch EPROMs ersetzt werden — 528 Mark
mit CMOS-RAMs und Batteriepuffern — 648 Mark

**Speichererweiterung (in Planung):**
64-KByte-RAM-Erweiterung, eventuell mit Centronics-Schnittstelle und Extended Basic. Die vollen 64 KByte sind dann allerdings nur in Maschinencode ansprechbar. Ebenfalls geplant ist eine 32-KByte-RAM-Erweiterung mit dynamischen RAMs.

**Kabel (erhältlich):**
für Kassettenrecorder — 48 Mark
V.24-Anschluß (2m) — 60 Mark

**Schnittstellen (erhältlich):**
V.24-Schnittstelle (entspricht RS232C)
mit Gehäuse und Netzteil, für Senden und Empfangen
mit 2 Ports — 448 Mark
mit einem Port — 398 Mark

**weiteres Zubehör (in Planung):**
Diskettencontroller für doppelseitige und doppelt dichte Aufzeichnung auf maximal 3 Laufwerken (Mai 84)
Diskettenlaufwerke 5½-Zoll und 3½-Zoll
EPROM—Brennerkarte zum Anschließen an die Centronics-Schnittstelle
separate Centronics-Schnittstelle

Info: Atronic, Postfach 700124, 2000 Hamburg 70, Telefon (040) 6682256.

## Hurra — noch lange kein Ende in Sicht

Einige kleinere Zubehörteile können übrigens von Bastlern mit Löterfahrung billiger selbst gebaut werden. Dazu gehören Anschlußkabel und Adapter. Eine erste Hilfe bieten wir in der nächsten Ausgabe von Happy-Computer mit dem Bauplan für einen Joystickadapter. Weitere technische Unterlagen können zur Zeit von Texas Instruments in Form zweier Nummern der »Nachrichten«, einem Info-Blatt für TI 99/4A-Besitzer, angefordert werden. Unabhängig von Texas Instruments erscheint im Hasse-Verlag eine Anwenderzeitschrift mit Programmen und Hardwaretips, das »TI Magazin«. In Österreich erscheint eine Clubzeitschrift, »99 Journal«, mit sehr guten Beiträgen.

Zubehör aus den USA ist meist nicht für den unveränderten Anschluß geeignet. Wenigstens die Stromversorgung müßte den hiesigen Verhältnissen angepaßt werden. Das bedeutet im einfachsten Fall ein Auswechseln des Trafos. In vielen Fällen sind aber weitergehende Änderungen nötig, mit dem Risiko einer Beschädigung. Wir stellen hier deshalb ausschließlich Zubehörartikel vor, die von deutschen Fremdherstellern zur Zeit oder bald angeboten werden.

Die Angaben unserer Übersicht basieren auf Herstellerangaben und wurden nicht weiter überprüft. Sie erhebt keinen Anspruch auf Vollständigkeit. Wir werden aber auch in Zukunft die eine oder andere Peripherie für den TI 99/4A testen und kleine Selbstbauvorschläge veröffentlichen. (lg)

# Wie reinigt man d des TI 99/4A?

**Die Tastatur der ersten Serien des TI 99/4A weist nach einiger Zeit eine sehr unangenehme Eigenschaft auf: So kann es bei Betätigung verschiedener Tasten vorkommen, daß die gewünschten Zeichen gleich mehrmals hintereinander auf dem Bildschirm erscheinen.**

Dieses sogenannte »Prellen« muß man aber nicht hinnehmen. Wir zeigen hier, wie man es mit ein wenig Geschick beheben kann. Zur Technik der Tastenentprellung sei hier angemerkt, daß es eigentlich nur zwei Methoden gibt: Zum einen die softwaremäßige Entprellung und zum anderen — wie beim TI 99/4A praktiziert — die mechanische Entprellung. Bei der mechanischen Entprellung müssen hochwertige Kontakte eingesetzt werden, denn das Problem liegt darin, daß die Tastaturkontakte beim Schließen ins Schwingen geraten können und damit mehrmals hintereinander Kontakt geben. Für den Computer bedeutet dies aber: mehrmaliges Drücken der gleichen Taste. Das Problem wurde beim TI 99/4A durch eine ausgetüftelte Mechanik behoben. Man versah die beiden Kontaktfedern mit je zwei Kerben, vergoldete sie zwecks besserer Leitfähigkeit, und schützte den Kontaktblock mit einem Rahmen vor Staub. Bei noch neuen Geräten funktioniert dieses System relativ gut, werden sie aber älter, kann Oberflächenverschmutzung der Kontakte oder Verschleiß an den Plastikführungen und eine dadurch bedingte Schwergängigkeit einzelner Tasten zu Problemen führen. Um diesen Fehler zu beheben, muß man die Kontakte säubern.

Dazu muß jede Tastenkappe aus ihrer Verankerung entfernt werden. Bedenken Sie aber bitte: Sie sollten sich mit dem Reinigungsvorgang viel Zeit nehmen und große Vorsicht walten lassen. Die Demontage der Tastatur ist nicht ganz unproblematisch, da die Bauteile zum größten Teil aus Plastik bestehen und deshalb leicht zerbrechen können. Sollten Sie Schwierigkeiten haben — beispielsweise eine Taste will sich überhaupt nicht aus der Verankerung lösen lassen — verwenden Sie nicht zu viel Kraft, denn wo rohe Kräfte walten, fliegen Späne. Denken Sie daran, daß Ersatzteile sehr teuer sind.

## Hilfsmittel

Einige Hilfsmittel sollte man für diese Arbeit bereithalten:
1) einen möglichst breiten Schraubenzieher
2) einen Bogen Papier (wenn möglich Fließpapier)
3) eine Dose Kontaktspray
4) eine Dose Konservierungsspray.

Bei den Sprays ist auf Verträglichkeit mit dem Kunststoff der Konsole zu achten.

Die Reinigung der Tastatur sollte man nur dann vornehmen, wenn man das Gerät für mindestens eine Nacht nicht verwenden möchte, da mit flüssigen Reinigungsmitteln gearbeitet wird, die nach Anwendung erst vollständig verdunsten sollen. Achten Sie außerdem darauf, daß die Stromversorgung abgestellt ist, bevor Sie mit der Demontage beginnen. Sind diese Grundvoraussetzungen erfüllt, kann es losgehen.

## Die Demontage der Tastatur

Zuerst ist entweder die »CTRL«- oder die »FCNT«-Taste zu enfernen. Setzen Sie hierfür Ihren Schraubenzieher an der Unterkante einer dieser beiden Tasten an und hebeln Sie diese vorsichtig heraus. Als nächstes nehmen Sie sich die große Leer-Taste vor. Beachten Sie bitte: Diese Taste ist an den beiden Enden in einen Metallbügel eingehängt. Verwenden Sie deshalb auch hier nicht zu viel Kraft, wenn sie sich nicht gleich aus ihrer Verankerung löst. Setzen Sie wieder den Schraubenzieher ungefähr in der Mitte an und hebeln Sie die Taste langsam heraus. Schieben Sie diese anschließend ganz nach links oder rechts (je nachdem, welche der beiden danebenliegenden Tasten zuerst entfernt wurde). Biegen Sie dann mit dem Schraubenzieher den Bügel soweit nach außen, bis dieser aus der auf der Unterseite angebrachten Öse ausgerastet ist.

Damit haben Sie die schwierigste

# die Tastatur

Bild 1. Detailfoto eines Tastenkontaktes

Phase auch schon überstanden, denn nun ist eine ganze Reihe von Tastenkappen zugänglich, die nur noch, wie oben beschrieben, ausgebaut werden müssen. Man sollte aber bedenken, daß die Kappen nach der Reinigung auch wieder montiert werden müssen. Aus diesem Grund sollte man sie ordentlich und in der Reihenfolge wie sie entfernt werden auf eine Arbeitsfläche legen. Sind alle Tastenkappen ausgebaut, werden Sie Ihren TI 99/4A nicht mehr wiedererkennen.

Sprühen Sie etwas Kontaktspray auf jeden der jetzt zugänglichen Metallkontakte. Hierfür sollten Sie das mit der Sprühdose mitgelieferte Plastikröhrchen verwenden, um das Mittel nur dort zu verteilen, wo es auch tatsächlich hin soll, nämlich auf die Kontakte. Lassen Sie das Spray jetzt etwa fünf Minuten einwirken, damit der volle Reinigungseffekt zur Geltung kommt. Die Wartezeit kann man nutzen, indem man sich ein etwa 15 Millimeter breites und 30 Millimeter langes Rechteck vom Fließpapier-Bogen abschneidet und auf eine Breite von etwa fünf Millimeter zusammenfaltet. Tränken Sie dieses Stück mit dem gleichen Reinigungsmittel.

Für den nun folgenden Arbeitsgang braucht man zwei ruhige Hände, denn das Papier muß zwischen die Kontaktfedern geschoben werden, wenn man den größten Reinigungseffekt erzielen will.

Schauen wir uns zunächst einmal einen der Kontakte genauer an (siehe Bild 1, 2), bevor wir uns an die Arbeit machen: Um jeden einzelnen Kontakt ist ein weißer, nach unten beweglicher Rahmen zu sehen. In der Mitte befindet sich ein Steg, der in Ruhestellung die beiden Kontakthälften auseinander drückt. Wird nun der weiße Teil nach unten bewegt, wandert der Steg mit und eine elektrische Verbindung entsteht. Soweit die Funktionsweise. Durch vorsichtiges Auf- und Abbewegen des Papierstreifens zwischen den Kontaktfedern können Sie auch festsitzenden Schmutz entfernen. Dieses Verfahren kann bei allen Tasten angewandt werden. Zum Schluß sollte nur noch eine kleine Menge des Konservierungssprays auf die Kontaktflächen gesprüht werden.

## Die Montage der Tastatur

Achten Sie bitte darauf, daß Sie die Tastenkappen korrekt und auf die richtigen Kontakte aufsetzen, denn sonst erkennen Sie später Ihre eigenen Eingaben nicht wieder. Sind Sie sich einmal über die Plazierung einer Taste nicht mehr sicher, dann nehmen Sie sich bitte eine Abbildung, zum Beispiel die Bedienungsanleitung, zur Hand, um die richtige Stelle zu finden.

Montieren Sie zuerst die Leer-Taste, die, bevor sie fest in ihren Platz gedrückt wird, wieder in den Führungsbügel eingehängt werden muß. Suchen Sie, durch vorsichtige seitliche Bewegungen, die Vertiefung auf der Unterseite der Taste, in die der weiße Kolben einrasten muß. Haben Sie die richtige Stelle ausgemacht, müssen Sie die Taste nur noch bis zum Anschlag in ihre Ausgangsposition drücken.

Wenn Sie Ihren TI 99/4A wieder in Betrieb nehmen, werden Sie eine angenehme Überraschung erleben: Die Tasten lassen sich »wie frisch geölt« bedienen und das leidige Prellen ist verschwunden. (wb)

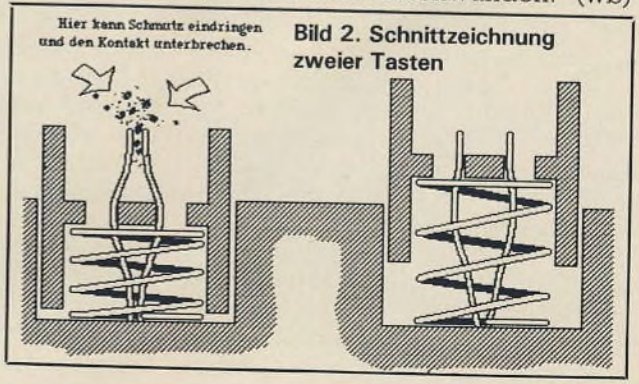



Bild 2. Schnittzeichnung zweier Tasten

# Maxi-Möglichkeiten mit dem Minimem-Modul?

**Im Unterschied zu den meisten Homecomputern besitzt der TI99/4A in der Grundversion keine Befehle, die den PEEKs und POKEs anderer Computer entsprechen. Da aber das TI-Basic ärgerlich langsam ist und die vorhandenen Möglichkeiten, andere Bildschirmformate betreffend, von dieser Sprachebene aus auch nicht zugänglich sind, werden viele Anwender bald den Wunsch verspüren, Maschinenprogramme erstellen zu können.**

Die drei möglichen Wege zur Maschinenprogrammierung erfordern in jedem Fall finanziellen Einsatz. Den billigsten Weg — ohne 32 KByte-Speichererweiterung und Diskettenlaufwerk — stellt das Minimem-Modul dar.

Dieses Modul mit dem ein auf Kassette gespeicherter Line-by-Line-Assembler und das im Bit-map-Mode geschriebenes Demonstrationsprogramm LINES geliefert wird, beinhaltet ein frei programmierbares, batteriegepuffertes RAM im Adreßbereich >7000 bis >7FFF. Der gleichfalls vorhandene Bereich >6000 bis >7000 gehört zu einem ROM mit Utilityroutinen.

Das im Minimem zur Verfügung stehende Basic deckt sich mit dem bekannten TI-Basic, ist aber erweitert um die Befehle

- CALL PEEK und CALL LOAD (dem üblichen PEEK/POKE entsprechend),
- CALL PEEKV/POKEV (für das VDP-RAM)
- CALL INIT (initialisiert den Speicherbereich) und
- CALL LINK (womit Maschinenroutinen in ein Basic-Programm eingebunden werden können).
- CALL LINK gestattet in sehr komfortabler Weise Parameterübergaben zwischen Basic- und Maschinenprogrammen.

Seitens Texas Instruments wurde das Minimem mit dem Hinweis angeboten, dies sei der preiswerteste Einstieg in die Welt der Maschinensprache. Bei vielen war die Enttäuschung dann aber groß, als sie in der — auch mit »dürftig« noch schmeichelhaft bewerteten beiliegenden Gebrauchsanweisung — kaum Hilfestellung bei der Erstellung eigener Programme fanden. Im Gegenteil, der Hinweis, für die Benutzung seien Assemblerkenntnisse Voraussetzung, führt die erwähnte Werbeaussage geradezu ad absurdum. Anders als beim Editor/Assembler-Modul, dessen mitgeliefertes Demonstrationsprogramm TOMBSTONE-CITY mit einem ausführlich kommentierten Source-Code versehen ist, gibt es zu LINES keine Information. Das ist um so bedauerlicher, als dieses, in der beim TI99/4A höchstverfügbaren Grafikbetriebsart geschriebene Programm, bereits alle Routinen enthält, die für den eigenen Aufbau eines Plotprogramms notwendig sind. Darüber hinaus wären von diesem Programm her einige Fehler des großen Systemhandbuchs zum Editor/Assembler (die Werte der Videoregister betreffend) korrigierbar. Unverzeihlich sind auch einige über das Maß von Druckfehlern weit hinausgehende fachliche Fehler in der ohnehin unzureichenden Minimem-Dokumentation.

## TMS 9900-Befehlssatz

Welche Möglichkeiten der Programmerstellung bietet nun der Minimem-Assembler? Der Assembler unterstützt bis auf wenige Ausnahmen den kompletten TMS9900-Befehlssatz, unterscheidet sich aber in einigen Punkten stark von den Möglichkeiten des Assembler/Editors, so daß auf letzterem entwickelte Programme nicht unbesehen eingetippt werden können.

Einer der Hauptunterschiede liegt in dem zu erstellenden Programmtyp. Am Minimem sind nämlich ausschließlich Programme mit absoluten Adressen möglich, während die des Editor-Assembler-Moduls — sofern nicht anders gewünscht — relocatibel erstellt werden. Damit entfällt am Minimem bereits die Programmdefinition DEF; der Pointer auf das Programm muß ohnehin vom Programmierer erstellt werden (was dem Anfänger durch die unzureichende Erklärung sehr schwer fällt). Schwierigkeiten gibt es auch mit den Labeln innerhalb eines Programms. Werden vom großen Assembler Label beliebigen Inhalts akzeptiert, so nimmt der Minimem-Assembler nur die letzten beiden Buchstaben des Labels an und meldet unbarmherzig »ERROR«, falls zum Beispiel »INPUT« und »OUTPUT« gleichzeitig verwendet werden. Ein weiteres Problem mit den Labeln ergibt sich aus der Lage der Referenztabelle, da diese bereits nach zehn verwendeten Labeln von der normalen Programmstartadresse >7D00 aufwärts ein eventuell bestehendes Programm zu überschreiben beginnen.

Überhaupt ist eine Schwierigkeit bei der Erstellung längerer Programme der Assembler selbst, der seinerseits einen großen Teil des RAMs belegt. Bei unachtsamer Adressenwahl wird er zerstört und muß neu geladen werden. Natürlich kann durch trickreiches Programmieren das gesamte RAM genutzt werden. Dann ist es aber schon fast zeitsparender, die Programme gleich mittels eines in Basic geschriebenen Assemblers ins RAM zu POKEn.

Ein weiterer Unterschied zum Editor/Assembler ist das Fehlen der Direktive REF, die durch EQU ersetzt werden muß. Nicht akzeptiert wird BYTE, das unter Verlust eines Bytes aber durch DATA nachgebildet werden kann. Außerdem wird RT (Return) nicht angenommen. Dieser Befehl kann aber durch das Opcode-gleiche »B *R11« ersetzt werden.

Da der Line-by-Line-Assembler keine Möglichkeit eines Ausdrucks des eben eingegebenen Source-Codes bietet, ist weder die Nutzung eines Kommentarfeldes vorgesehen, noch werden die Editiermöglichkeiten ausreichend unterstützt. Das Fehlen eines Zeileneinschub- oder Zeilenlöschbefehls macht sich bei Programmumarbeitungen sehr nachteilig bemerkbar, da der Zeileneinschub nur durch Neueingabe ab der betreffenden Adresse durchführbar ist (wobei bereits in der alten Version benutzte Label nicht mehr verwendet werden kön-

nen). Das Entfernen von Zeilen ist nur durch Umwandlung in einen NOP-Befehl möglich. Welche Editiermöglichkeit besteht nun wirklich? Mittels der Pfeiltasten FCTN »E« und »X« kann der Bildschirm nach oben oder unten gescrollt werden, wobei neun Schirmbilder gespeichert sind. Bei Programmen, die länger sind, erscheint wieder das erste Schirmbild, über dessen Inhalt das neue Programm geschrieben wird. Dabei geht zumeist die Übersichtlichkeit verloren.

Nicht unerwähnt bleiben darf im Zusammenhang mit den Editiermöglichkeiten die Möglichkeit des sogenannten »EASY BUGS«, womit byteweise der gesamte Speicherraum inspiziert und, soweit es sich um RAM handelt, abgeändert werden kann. Da dazu aber bereits eine tiefere Systemkenntnis erforderlich ist, bleibt dem Anfänger diese Möglichkeit zunächst sicher verschlossen.

## EASY-BUG für Kenner

Mittels Easy-Bug können Maschinenprogramme auch unmittelbar gestartet werden. Leider kann aber diese Option nicht zum Austesten von Unterprogrammen, die später mit einem Basicprogramm verbunden werden sollen, genutzt werden da keine Parameterübergabe möglich ist. Außerdem wird in den seltensten Fällen etwas am Bildschirm zu sehen sein, da bei Maschinenprogrammen, die mittels CALL LINK aus dem Basic aufgerufen werden, für die Bildschirmausgabe ein Screen-Offset von >60 bei allen ASCII-Codes zu berücksichtigen ist.

Erfreulicherweise ist die Laderoutine für Maschinenprogramme mit der des Editor/Assemblers kompatibel, so daß am »großen Bruder« entwickelte Maschinenprogramme mit maximal 4 KByte Länge ins Minimem geladen und ausgeführt werden können, was man vom Loader des Extended-Basic-Moduls nicht behaupten kann. Unter Berücksichtigung der anderen Utility-Adressen für Extended Basic können aber mit dem Minimem (und einer vorhandenen RAM-Expansion) auch Programme für Extended Basic im Bereich >24F4 bis >3FFF geschrieben werden, da diese beim Modulwechsel erhalten bleiben.

Unerklärlich bleibt freilich, warum Texas Instruments dem Minimem keine der Diskettenladeroutine äquivalente Speicherroutine gegönnt hat. Alle Programme können mit der SAVE-Option des Easy-Bug nur auf Kassette gespeichert werden und fallen somit für Extended Basic wieder aus, da diesem die Laderoutine fehlt. Es macht zwar keine große Mühe, sich das entsprechende Maschinenprogramm selbst zu schreiben, doch wird damit immer unnötigerweise ein Teil des RAMs blockiert, das mit dem Assembler geteilt werden muß.

Auch eine RESET-Taste hätte dem Minimem-Modul nichts geschadet, da diese am TI99/4A nicht vorhanden ist und der angehende Maschinenprogrammierer häufig die Bekanntschaft von »Abstürzen« macht. In dieser Phase wird der Ein-/Aus-Schalter ohne solche Taste über Gebühr stark beansprucht.

Haben wir bisher kräftig kritisiert (wie sich das nach intensiver Anwendung des Moduls aufgrund der genannten Eigenschaften zwangsläufig ergeben muß), so sollen jetzt die guten Eigenschaften gebührend hervorgehoben werden. Beim Grundsystem ohne RAM-Expansion und Diskettenlaufwerk ist es höchst angenehm, bestimmte Unterroutinen permanent im Minimem verfügbar zu haben (zum Beispiel echte Kleinbuchstaben, Grafik- und Hardcopyroutinen etc.). Ja selbst Basic-Programme bis 4 KByte Länge können mit »SAVE "MINIMEM"« dauerhaft gespeichert werden (zerstören damit aber eventuell Maschinenprogramme).

Verglichen mit den Investitionen, die das große Editor/Assembler-Modul verlangt (praktisch Vollausbau mit RAM-Expansion und Diskettenlaufwerk), fällt das Minimem durch ein hervorragendes Preis/Leistungsverhältnis auf.

Nach einer Einarbeitungsphase und vor allem dem Erwerb weiterführender Literatur (hier speziell das auch einzeln verkaufte Handbuch zum Editor/Assembler), steht einem sicherlich die gesamte Welt der Maschinensprache am TI99/4A offen und der Hemmschuh TI-Basic kann überwunden werden. Einige hervorragende kommerzielle Programme mit ausgezeichneten Grafikmöglichkeiten unterstreichen dies ebenso, wie die Tatsache, daß mit dem Minimem auch Forth zugänglich wird.

Leider existiert zum TMS9900 keine Programmbibliothek, die denjenigen für die Prozessoren Z80 oder 6502 vergleichbar wäre. Auch die Zahl der Veröffentlichungen mit TMS9900-Maschinenprogrammen in Zeitschriften ist noch leicht überschaubar. Gerade aber durch die Bemühungen diverser 99-Userclubs, einführende Literatur anzubieten, steigt die Anzahl der Maschinenprogrammierer, und es ist kurios genug, daß der Höhepunkt dieser Entwicklung sicher erst ein, zwei Jahre nach der Produktionseinstellung des TI99/4A eintreten wird. Bremsend auf diese Entwicklung wirkt sich aber der Umstand aus, daß noch immer keine Beschreibung des Betriebssystems existiert und die Kenntnisse wichtiger Systemadressen der Maschine erst mühsam abgerungen oder von Freunden ertauscht werden müssen.

Da aber mit dem Minimem auch dem Einsteiger sehr bald die Fähigkeiten des Computers offenbar werden (zum Beispiel Hochzählen mit Bildschirmanzeige von 0 auf 10000 in 0,8 s, Grafik im Bit-map-Mode mit 256 x 192 Pixels mit fast unbegrenzter Farbsetzmöglichkeit, Textmode mit 40-Zeichen/Zeile, Tonverarbeitung mit Echo und Hall, 32 Sprites, direkte Sprachausgabe und so weiter), kann das Modul trotz der geschilderten Mängel nur wärmstens empfohlen werden.

## Mit Minimem mehr wert

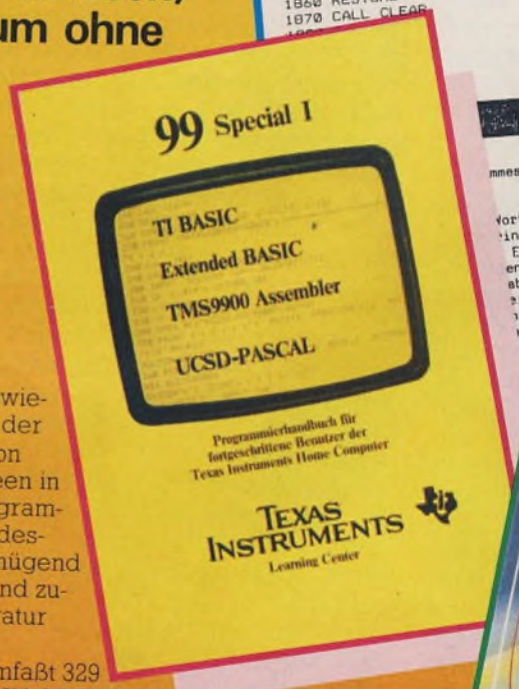
Ist eine 32-KByte-RAM-Erweiterung vorhanden, dann steht am TI99/4A auch der größtmögliche RAM-Bereich von insgesamt 36 KByte für Maschinenprogramme zur Verfügung, wobei das Minimem auch ein File »EXPMEM2« (der RAM-Bereich >A000 bis >FFFF) als Daten- und Programmspeicher — ähnlich einer RAM-Disk — unterstützt. Da durch die Batteriepufferung darüber hinaus eine rasche Portabilität von Programmen zwischen verschiedenen Geräten besteht, wird die Abhängigkeit vom Kassettenrecorder geringer. Ebenso findet die mit dem Recorder am TI99/4A freudlose Dateibearbeitung durch das File »MINIMEM«, das nur einmal geladen und bei Ende des Programms gesichert wird, für kleine Datenmengen eine Ergänzung, die viel bequemer ist und alle Möglichkeiten eines Diskettenfiles besitzt (zum Beispiel die Befehle UPDATE, RECORD#, RESTORE#).

Der Kauf des Moduls erhöht den Gebrauchswert des TI 99/4A erheblich und es ist zu hoffen, daß sich ähnlich wie für das Extended-Basic-Modul, Fremdhersteller finden, die das Minimem-Modul in Zukunft anbieten werden.

(Karl Hagenbuchner)

# Tausendundeine Seite Lesestoff...

**Es ist noch kein Meister vom Himmel gefallen, auch kein Meisterprogrammierer. Bücher über das Programmieren zu lesen, ist der einfachste Weg, um ohne große Umwege wenigstens die Grundzüge zu erlernen. Wir haben für Benutzer des TI 99/4A einige Buchbesprechungen in Kurzform zusammengestellt.**



**TI-Basic/Extended-Basic für Anfänger und Fortgeschrittene**

Dieses Handbuch ist nahezu ein Muß für alle, die neben der Konsole noch das Extended-Basic-Modul besitzen. Es enthält eine vollständige und ausführliche Beschreibung aller Basic- und Extended-Basic-Befehle. Während das offizielle Handbuch zum Extended-Basic nur in Englisch erhältlich ist, vermittelt der vorliegende Band den gleichen Stoff auf Deutsch und darüber hinaus noch viele weitere nützliche Informationen. Wer ihn besitzt, kann das ursprüngliche Handbuch vergessen.

Was er dem Leser aber kaum vermittelt, ist Programmiertechnik und das Wissen, wie man mit Programmen umgeht. Anders ausgedrückt: Dieses Buch ist kein Lehrbuch (der erste Teil ist nur eine Wiedergabe der blauen Begleitbroschüre zur Konsole), sondern ein Nachschlagewerk. Besonders wer Schwierigkeiten bei der Umsetzung von Programmideen in konkrete Programme hat, wird deshalb nicht genügend Hilfe finden und zusätzliche Literatur benötigen.

Das Buch umfaßt 329 Seiten im DIN-A5-Format, kostet 48 Mark und bietet für seinen Preis hochkonzentrierte Information (hervorragendes Preis-/Leistungsverhältnis). Es ist keine unterhaltsame Kost, sondern ein notwendiges Werkzeug.

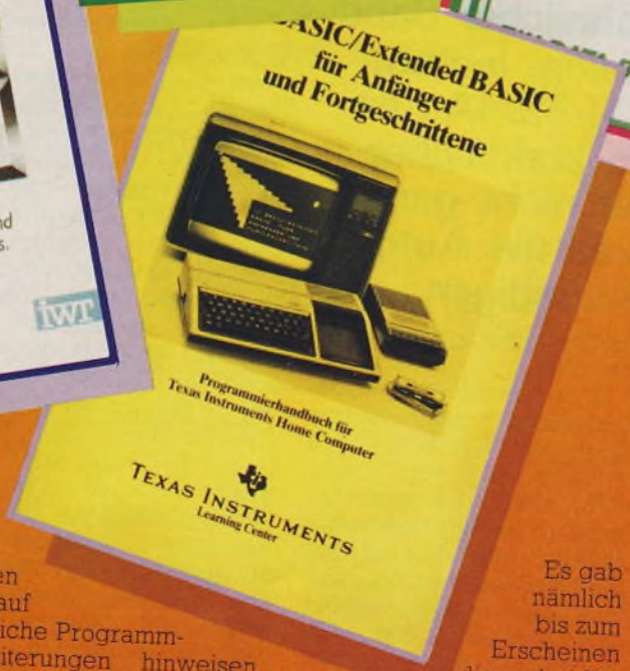
Paperback von Texas Instruments Learning Center, Freising, ISBN 3-88078-039-0

**TI 99, Tips & Tricks**

Als »eine Fundgrube für den TI-99-Anwender« bezeichnet der Autor Karl P. Schwinn sein Buch. Diese durchaus berechtigte Bezeichnung deutet schon an, was das Buch nicht sein will und was es leisten möchte: Es erhebt nicht den Anspruch, den Leser vom Nullpunkt aus zum Fachmann machen zu können. Es möchte dem Leser vielmehr über das Handbuch hinaus zusätzliche Tips, Tricks und weiterführende Anregungen für eigene Programme geben. Dazu bietet es gleich zu Beginn einen nützlichen Abschnitt, in dem wichtige Befehle des TI-Basic mit entsprechenden Microsoft-Basic-Befehlen verglichen und Hinweise zur Konvertierung gegeben werden. Damit kann der Leser bei einigem Geschick MBasic-Programme für den TI 99/4A umstricken.

Viele Beispiele und Hinweise beziehen sich auf das Extended-Basic. Wer dieses nicht besitzt, kann das Buch natürlich auch mit Gewinn lesen, wird aber nicht alles darin gebotene in die Praxis umsetzen können.

Das Buch besitzt 300 Seiten im DIN-A5-Format, kostet 49




Grundausstattung, also der Konsole ohne Erweiterungen, gespielt werden können. Einführende Kapitel erläutern die wichtigsten CALL-Unterprogramme. Jedem Spiel sind außerdem Erläuterungen vorangestellt, die in knapper Form die Bedienung erklären und auf mögliche Programmerweiterungen hinweisen. Die 20 Spiele sind gut gemixt und umfassen sowohl Action- wie auch Denkspiele. Eine ausführliche Besprechung ist in Happy-Computer, Ausgabe 3/März 1984 auf Seite 136 erschienen.

Das Buch kostet zirka 38 Mark, ist 217 Seiten stark und ist vor allem für jene zu empfehlen, die nur die Konsole ohne Erweiterungen besitzen. Für sie gibt es nämlich ohnehin kaum Literatur.

Paperback vom IWT-Verlag, Vaterstetten bei München, ISBN 3-88322-045-0

**99 Special I und II**

Die Autorin Dr. Alma Peschetz hat zweifellos eine eklatante Lücke im Literaturangebot für den TI 99/4A geschlossen. Es gab nämlich bis zum Erscheinen dieser beiden Bücher eigentlich keine deutschsprachige Fachliteratur für Programmiersprachen auf dem TI 99/4A außer derjenigen für Basic und Extended-Basic. »Special I« und »Special II« enthalten nun Programme und Informationen zu TI-Basic, Extended-Basic, TMS-9900-Assembler und UCSD-Pascal. Je nach Verbreitung bei den Anwendern ist auch die mengenmäßige Gewichtung im Buch vorgenommen worden. Die Anteile am Umfang reichen von 75 Prozent für Basic bis zu 10 Prozent für Pascal.

Zu jeder Sprache findet der Leser Programmlistings aus den Bereichen Mathematik, Grafik und Musik. Ein Teil dieser Programme ist sehr anspruchsvoll und setzt bereits eine ganze Menge Vorkenntnisse voraus. Besonders im Band II kommen Themen zur Sprache, die weit über den Heimcomputerbereich hinausgehen. Beide Bücher sind didaktisch gut gemacht und können dem erfahrenen Basic-Programmierer empfohlen werden.

Der Band I besitzt zirka 300 Seiten, kostet 49,50 und wurde bereits in Happy-Computer, Ausgabe 2/Februar 1984 auf Seite 84 ausführlich besprochen. Band II erscheint in diesen Tagen, kostet 54 Mark und umfaßt 480 Seiten.

Paperbacks von Texas Instruments, Freising, ISBN 3-88078-043-9 (Band I), ISBN 3-88078-045-5 (Band II)

**16-Bit-Mikroprozessor-Kursbuch**

Schon etwas betagter ist dieses Lehrbuch für den Maschinencode des TMS 9980A, einer kompatiblen CPU zum TMS 9900 des TI

Mark und setzt den Wissensstand des Handbuchs voraus. Vor allem die abgedruckten Programmbeispiele sind das Lesen wert. Eine ausführliche Besprechung ist in der Zeitschrift Computer persönlich, Ausgabe 1/84 auf Seite 123 erschienen.

Paperback von Data Becker, Düsseldorf, ISBN 3-89011-006-1

**TI 99/4A, Farbe, Grafik, Ton, Spiele**

Dieser Band von Guido Pahlberg enthält 20 Spiele, die ausnahmslos mit der

## Tausendundeine Seite

# Lesestoff...

99/4A. Genau genommen behandelt das Buch das Mikroprozessor-Lehrsystem TM990/189. Wer deshalb das Gelesene auf den TI 99/4A anwenden will, muß schon sehr weit in die Kunst des Programmierens eingedrungen sein. Eines ist sicher: Mit diesem Buch allein kann niemand Assemblerprogramme für den TI 99/4A schreiben. Da andererseits viele Besitzer des Editor/Assemblers oder des Minimem-Modus nur ein englischsprachiges Handbuch besitzen, kann dieses Buch in solchen Fällen komplizierte Sachverhalte in deutsch durchschaubar machen.

Zwei Kapitel zum modularen Programmieren und zu professioneller Softwareentwicklung sind auch für Hobbyprogrammierer interessant. Das Buch besteht aus 463 Seiten im DIN-A5-Format und kostet 32,50 Mark.

Paperback von Texas Instruments, Freising, ISBN 3-88078-029-3

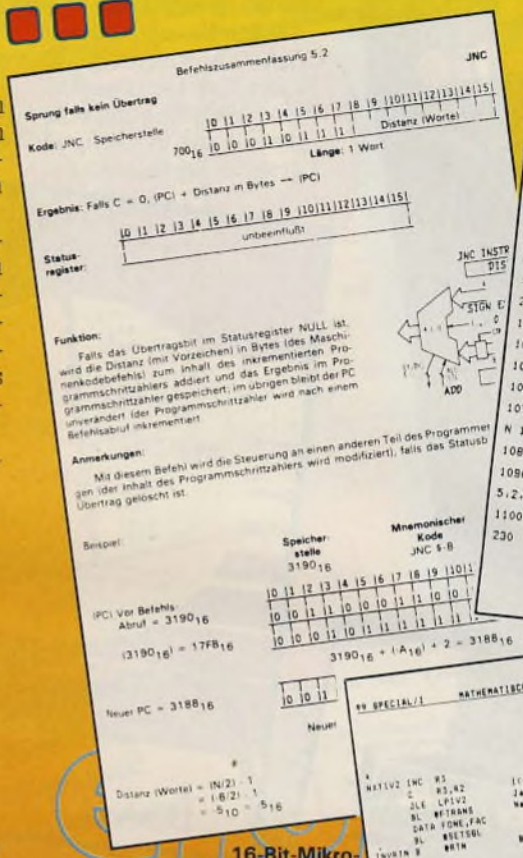
# Mehr arbeiten als spielen mit dem TI 99/4A

**Ein etwas ungewöhnlicher Band mit dem Titel »spielen, lernen, arbeiten mit dem TI-99/4A« ist erschienen, dessen praktischer Wert groß ist, aber nicht ganz dem entspricht, was die Autoren im Vorwort ankündigen.**

Im Vorwort wird dem Leser versprochen, er könne mit dem Buch auf spielerische Weise solides EDV-Handwerkszeug und die Fähigkeit zur Anpassung fremder Programme an die eigenen Bedürfnisse erwerben. Unter einem soliden EDV-Handwerkszeug ist aber sicher nicht die Beherrschung eines einzelnen speziellen Computers (noch dazu eines Heimcomputers zu verstehen). Ohne Zweifel: Wer mit dem TI 99/4A umgehen kann, hat einen leisen Vorgeschmack dessen bekommen, was Computer können — aber eben nur einen Vorgeschmack. Rüstzeug für die Anpassung fremder Programme wird darüber hinaus fast keines gegeben.

Andererseits: Eine ausgesprochene Stärke des Buchs wird überhaupt nicht erwähnt und vom Leser wohl erst dann bemerkt, wenn er im Inhaltsverzeichnis das große Kapitel »Arbeiten« aufmerksam studiert oder die ersten 110 Seiten durchgeblättert hat. Ich meine diejenigen Abschnitte, in denen eine Menge ausgezeichnete bankkaufmännische und andere Finanzprogramme zum Eintippen vorgestellt werden. Man kann sich des Eindrucks nicht erwehren, daß die Autoren in der Bankenbranche beheimatet sind.

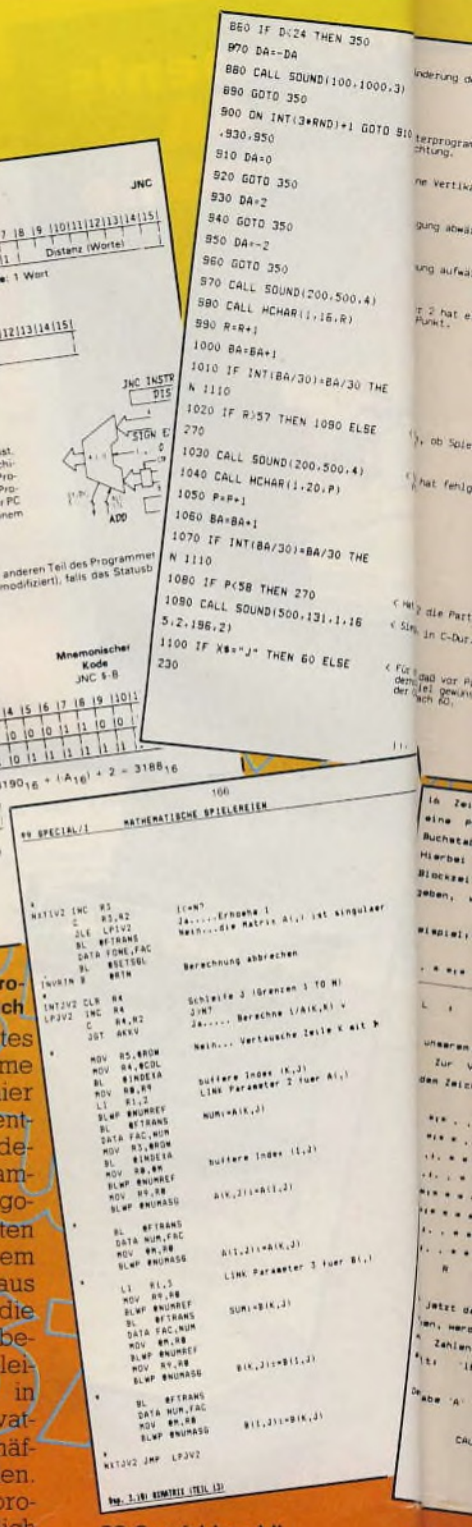
Eine Reihe dieser im




16-Bit-Mikro-Prozessor-Kursbuch

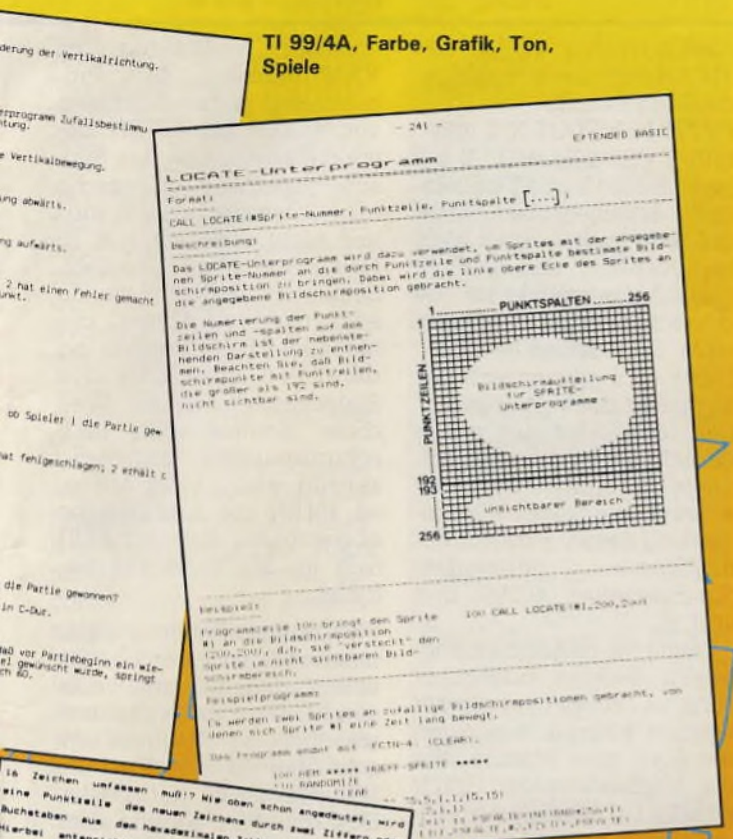
wahrsten Sinn des Wortes »wertvollen« Programme dürften in dieser Form hier zum ersten Mal veröffentlicht worden sein. Besondere Stärke dieser Programme sind die Lösungs-Algorithmen. Die verwendeten Formeln entsprechen dem Banken-Standard. Daraus läßt sich übrigens auch die Eignung für einen ganz besonderen Leserkreis ableiten: Kundenberater in Geldinstituten und Privatleute, die ihre Bankgeschäfte »im Griff« haben wollen. Mehrere Arten von Zinsprogrammen (einschließlich Zinsstaffel), Kalkulations- und Investitionsprogramme sowie ein Währungsumrechnungsprogramm sind vertreten. Nebenbei kommen in diesem sehr ergiebigen Kapitel die Menütechnik, ein Kalender- und ein Balkendiagrammprogramm zur Sprache — sinnvolle Ergänzungen zu den oben genannten Programmen.

Das Kapitel »Lernen« ist das kürzeste und gibt dem

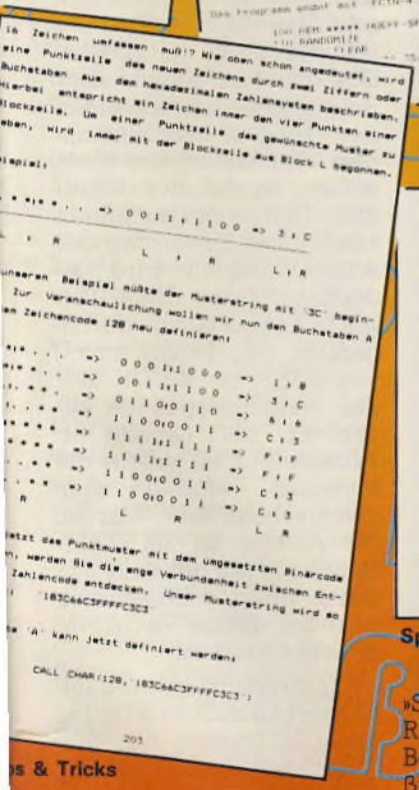



99 Special I und II

Leser nur ansatzweise Informationen, allerdings von einer Art, die in den anderen Büchern zum TI 99/4A normalerweise nicht einmal erwähnt werden. So erfährt der Leser etwas über den internen Aufbau der Hardware im TI 99/4A, die Organisation der CPU und die Speicheraufteilung. Ergänzt wird dieses Kapitel durch ein kurzes Programm zum



**TI 99/4A, Farbe, Grafik, Ton, Spiele**



**TI-Basic/Extended-Basic für Anfänger und Fortgeschrittene**

```
1000 REM ======================
1010 REM  UMWANDLUNGSPROGRAMM
1020 REM  OKTAL/DEZIMAL/HEXAD
1030 REM     AUS BINAERZAHLEN
1040 REM ======================
1050 REM HAUPTROUTINE
1060 ON WARNING NEXT
1070 GOSUB 1340
1080 FOR I=1 TO 10
1090 DB=0
1100 ACCEPT AT(I+4,1)VALIDATE("01")SIZE(8):B$
1110 IF LEN(B$)=0 THEN GOTO 1100
1120 GOSUB 1230
1130 REM
1140 DISPLAY AT(I+4,18):USING "###":DB
1150 DISPLAY AT(I+4,25):H1$
1160 DISPLAY AT(I+4,28):H2$
1170 NEXT I
1180 GOSUB 1420
1190 GOTO 1050
1200 REM **********************
```

**Spielen, lernen, arbeiten mit dem TI 99/4A**

**s & Tricks**

Auslesen des Speichers und eine Liste wichtiger Systemadressen im Anhang. Nicht, daß der Leser damit schon in Maschinensprache programmieren könnte. Aber es befriedigt doch einiges von der gesunden Neugier des Einsteigers.

Am irreführendsten ist die Bezeichnung des Kapitels »Spielen«. Zwar sind einige Routinen aufgelistet, deren Bedeutung über einen mäßig »spielerischen« Selbstzweck nicht hinausgeht (von einem Orgel-Programm vielleicht einmal abgesehen), der eigentliche Zweck scheint aber die Vermittlung einiger Basisinformationen zu den Programmiermöglichkeiten des TI 99/4A zu sein. Erwähnenswert ist ein kurzer Abschnitt über die Erzeugung einer Tonleiter mit Hilfe des Sound-Befehls. Die Autoren geben dafür nämlich einen echten Algorithmus an und beschreiiben nicht die speicheraufwendige schwerfällige Allerweltsmethode mit den diskret abgespeicherten Frequenzwerten.

Bevor wir uns allgemeineren Eigenschaften des Buchs zuwenden, noch ein Lob für den Anhangteil. Neben den schon erwähnten Systemadressen findet der Leser dort unter anderem Listen mit dem Maschinencode und den Mnemonics des TMS 9900 (die CPU des TI 99/4A), mit den Codierungen des Standardzeichensatzes (nützlich, wenn eigene Zeichen entworfen werden sollen) und den Befehlen des Extended Basic (interessant für jene Leser, die noch kein Extended Basic-Modul besitzen, sich aber mit dem Gedanken tragen, eines zu kaufen).

Für einen Teil der abgedruckten Programme ist allerdings dieses Extended Basic-Modul Voraussetzung. Die Autoren weisen im Einführungsteil darauf hin. Weitere Zusätze zur Konsole sind aber nicht nötig. Die Wahl der begrenzt erweiterten Konfiguration ist gerechtfertigt. Einerseits stellt das Extended Basic-Modul nämlich eine nur mäßig teure Erweiterung dar, andererseits ist der Gewinn an zusätzlichen Möglichkeiten beim Programmieren enorm.

Die Autoren, K.-J. Schmidt und G.-P. Raabe, setzen bei ihren Programmen glücklicherweise keine Diskettenstation und keinen Drucker voraus. Leider fehlen Hardcopys oder Bildschirmfotos gänzlich und damit sogar dort, wo sie inhaltlich wichtig wären, zum Beispiel auf jenen Seiten, auf denen die Farbmöglichkeiten erläutert werden.

Insgesamt ist das Buch seinen Preis von 28 Mark wert. Auch wer schon Literatur zu seinem TI 99/4A besitzt, wird hier neue Anregungen und Informationen finden. Ohne ergänzende Literatur ist es für den Einsteiger allerdings weniger geeignet, da die Themen nicht grundlegend genug dargestellt werden. Am meisten wird der schon etwas geübte Basic-Programmierer von diesem Band profitieren können.(lg)

K.-J. Schmidt, G.-P. Raabe, »spielen, lernen, arbeiten mit dem TI-99/4A«, Sybex, ISBN 3-88745-039

## Texas Instruments TI 99/4A Programme

Mehr ein besonders dickes Heft als ein Buch ist der Chip-Sonderband für den TI 99/4A. Im Gegensatz zu den bisher besprochenen Büchern bietet dieser Sonderband ausschließlich Programme mit Programmbeschreibungen, aber keine Erläuterungen oder Hilfen für den Anfänger.

Dafür sind die Programmbeschreibungen ausführlich und gut. Bei den Programmen sind die Bedienung — und in einigen Fällen — die Bildschirmausgaben erläutert. Bei den Spielprogrammen kommen meist noch eine Beschreibung der Spielregeln und des Spielablaufs hinzu. Was der Leser aber bei diesen Programmen besonders vermissen wird, sind Bildschirmfotos. Die Listings sind gut leserlich, mit großer Schrift wiedergegeben.

Der Sonderband enthält auf knapp 100 Seiten 10 Spiele-, 5 Grafik- und 8 ernsthafte Programme. Alle Programme sind Erstveröffentlichungen. Es ist die preiswerteste unter den hier genannten Publikationen und kostet 18 Mark.

Sonderheft von Vogel-Verlag, Würzburg

# Fallobst für Freaks

## Vom Gemeindecomputer zum Listing des Monats

**Stefan Züger (links im Bild), Jahrgang 1967, und Erich Schnellmann (rechts auf dem Foto zu sehen), Jahrgang 1966, leben beide in der kleinen Schweizer Landgemeinde Wangen, nicht weit von Zürich entfernt.**

Für Stefan zählt neben der Computerei das Segeln auf dem Zürichsee zu seinen ganz besonderen Hobbies. Zur Zeit besucht er die vierte Klasse des Kollegium Nuolen, eine Art humanistisches Gymnasium. Wie seltsam oft die Wege sind, auf denen der erste Kontakt mit dem Computer zustande kommen kann, zeigt gerade seine Geschichte.

Die Gemeinde Wangen hat rund 3000 Einwohner. Stefans Vater, Gemeindekassier in Wangen, bekam eines Tages für seine Arbeit einen Cobol-Computer zur Verfügung gestellt. Stefan wurde neugierig und versuchte dessen Betriebssystem zu ergründen. Ein hoffnungsloses Unterfangen für einen Computerlaien, der Stefan damals noch war. Immerhin weckte die Beschäftigung mit dem professionellen System in ihm den Wunsch nach einem eigenen, beherrschbaren, Computer. Von einem Bekannten erstand er deshalb einen PET, dessen Möglichkeiten aber bald den Ansprüchen nicht mehr genügten. Als sich sein Freund und Nachbar Erich, durch Stefan angesteckt, einen TI 99/4A kaufte, begann Stefan auf dessen Computer zu programmieren.

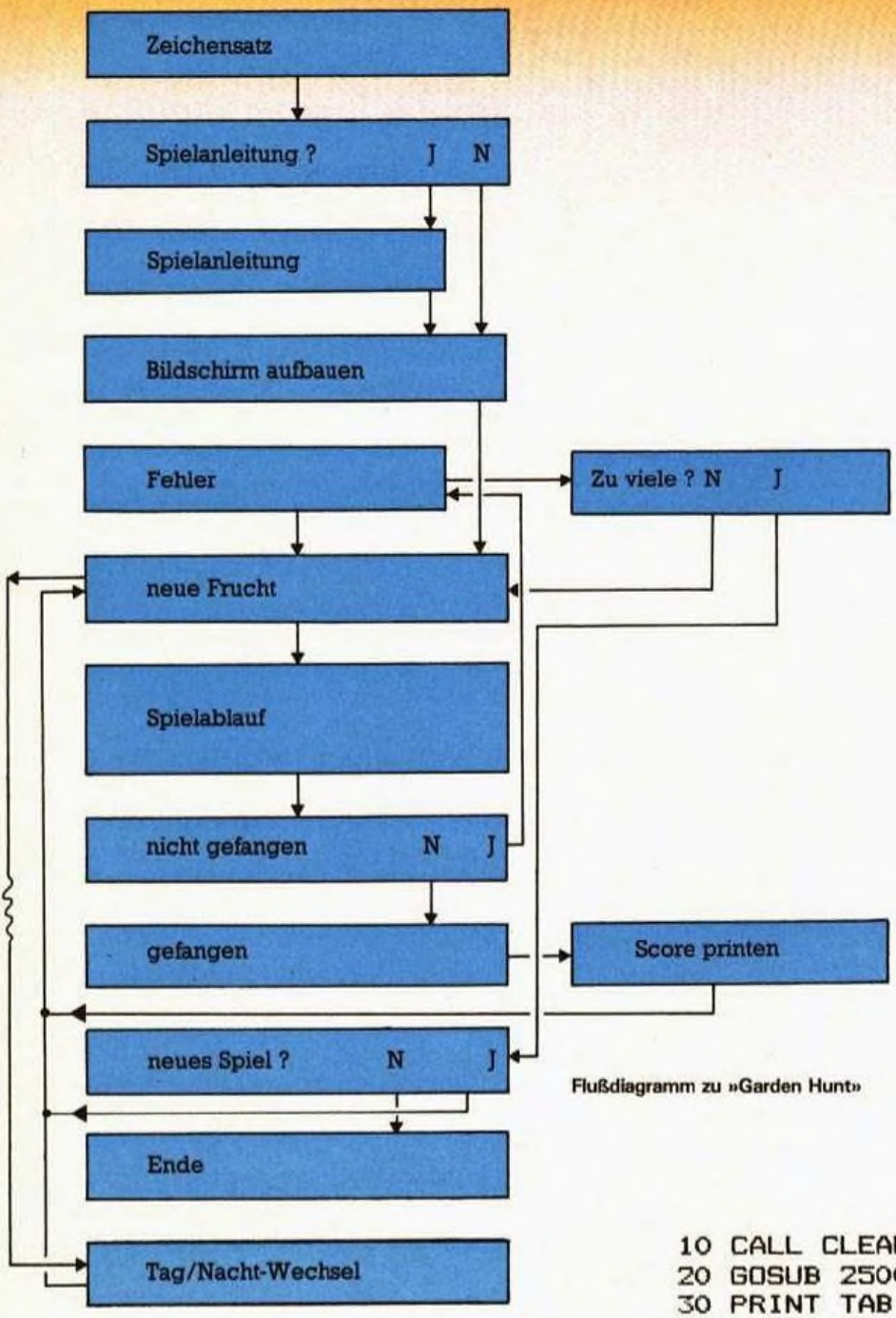
Zusammen schrieben sie in rund zwei Monaten das Programm »Garden Hunt«. Die Spielidee kam ihnen in einem Dorfgasthaus, in dem ein alter mechanischer Spielautomat steht. Im Automaten fallen von oben Kugeln durch ein Gaußsches Nagelbrett (an manchen Schulen dient ein solches Monstrum noch zur Demonstration der Gaußschen Verteilung), die am unteren Ende durch den Spieler mit Hilfe einer Clownsfigur aufgefangen werden müssen.

### Teamwork auch am Computer erfolgreich

Stefan hat beruflich mit EDV wenig im Sinn. Er plant, später einmal Journalismus zu studieren. Er leugnet jedoch nicht, zeitweise einer Art Computersucht verfallen zu sein, die er aber inzwischen überwunden hat. Dennoch will er sich von seinem Anteil an den 2000 Mark Prämie einen Commodore 64 kaufen. In diesem Punkt gleichen seine Pläne denen seines Freundes Erich, der allerdings mehr auf den Spectrum spekuliert. Erich Schnellmann, Azubi in Maschinentechnik, betreibt die Computerei im Gegensatz zu Stefan als einziges Hobby und kann sich durchaus vorstellen, später einmal seinen Beruf auf dieses Gebiet zu verlagern. Sein Engagement für die Computertechnik findet im Familienkreis jedenfalls volle Anerkennung.

Erstaunlich ist für uns, daß wieder einmal ein »Gemeinschafts«-Produkt das Rennen machte. Es scheint deutlich zu werden: Auch in der Computerei ist in der Regel Teamarbeit erfolgreicher als Einzelkämpfertum. Der Computer gibt uns Menschen also keinen Anlaß, der zwischenmenschlichen Isolation den Vorzug einzuräumen. Und noch etwas finden wir schön: Mit dieser Prämierung können wir den Beweis erbringen, daß wir unsere Leserschaft nicht an Ländergrenzen messen.

(lg)




Flußdiagramm zu »Garden Hunt«

```
10 CALL CLEAR
20 GOSUB 2500
30 PRINT TAB(6);"WUENSCHST DU EI
   NE": :TAB(3);"SPIELANLEITUNG
   ?  (J/N)": : : :
40 CALL KEY(0,K,S)
50 IF S=0 THEN 40
60 IF K=74 THEN 80
70 IF K=78 THEN 280
80 GOSUB 2690
90 REM
```

Listing zu »Garden Hunt«

# Fallobst für Freaks

```
100 REM********************
110 REM*                   *
120 REM*   GARDEN HUNT     *
130 REM*   -----------     *
140 REM*                   *
150 REM* (C) COPYRIGHTS    *
160 REM*        BY         *
170 REM*                   *
180 REM*ERICH SCHNELLMANN*
190 REM*BANHHOFSTRASSE 46*
200 REM*CH-8855 WANGEN     *
210 REM*                   *
220 REM*  STEFAN ZUEGER    *
230 REM*  GARTENSTRASSE    *
240 REM*  CH-8855 WANGEN   *
250 REM*                   *
260 REM*   13/01/1984      *
270 REM********************
280 CALL CLEAR
290 F1=0
300 SC=5
310 CALL SCREEN(1)
320 GOSUB 2050
330 MG=130
340 MOND=147
350 DATA 137,130,130,130,130,137
    ,130,130,130,137,152,153,154
    ,155,130,137,130,130,130,130
    ,130,130,137
360 DATA 130,130,130,137,130,130
    ,137,130,130
370 DATA 130,137,130,130,130,130
    ,130,137,130,130,156,157,158
    ,159,129,130,137,130,137,130
    ,130,137,130
380 DATA 130,130,137,130,152,153
    ,154,155,137
390 DATA 130,130,130,137,130,43,
    42,44,130,130,130,137,130,13
    0,130,137,130,130,43,41,42,4
    4,130,130,130,130,137
400 DATA 156,157,158,159,129
410 DATA 42,41,42,41,46,47,47,47
    ,45,44,130,137,130,43,41
420 DATA 41,42,46,47,47,47,47,45
    ,42,41,42,42,44,130,130,137,
    130
430 DATA 47,47,47,47,47,47,47,47
    ,47,47,45,42,46,47,47,40
440 DATA 47,47,47,47,47,47,47,47
    ,47,47,47,47,45,41,41,42
450 YS=16
460 PI(1)=72
470 PI(2)=88
480 PI(3)=80
490 PI(4)=73
500 CALL CLEAR
510 CALL CHAR(37,"183C3C7E7EFF7E
    18")
520 CALL CHAR(36,"0010387C7C1")
530 CALL CHAR(34,"0008081C08")
540 CALL CHAR(33,"00001038")
550 CALL CHAR(35,"000010387C1")
560 CALL CHAR(137,"0000000008")
570 CALL CHAR(148,"020303070F1EF
    C7")
580 CALL CHAR(147,"187EFFFFFFFF7
    E18")
590 CALL CHAR(72,"02041E3E3C7060
    8")
600 CALL CHAR(73,"08103C7A7E7E3C
    ")
610 CALL CHAR(74,"0073846414E3")
620 CALL CHAR(75,"009925252599")
630 CALL CHAR(76,"00CF28CE482F")
640 CALL CHAR(96,"182442ABD5AB56
    3C")
650 CALL CHAR(80,"78F010102844C3
    C3")
660 CALL CHAR(88,"80C0E06070381E
    07")
670 CALL CHAR(39,"0000001054381"
    )
680 CALL CHAR(38,"00000089523418
    1")
690 CALL CHAR(152,"000303000F1F3
    F3F")
700 CALL CHAR(153,"0000F0FFFFEFF
    FFF")
710 CALL CHAR(154,"000001F3FFFFF
    F3F")
720 CALL CHAR(155,"0000E0F0FC0E9
    EFE")
730 CALL CHAR(156,"7F6F6F6F3F16"
    )
740 CALL CHAR(157,"F8FFEF8FFFFF7
    B")
750 CALL CHAR(158,"19B3FFFFFFFF5
    9")
760 CALL CHAR(159,"FEFFEFFEFFF7C
    1")
770 CALL CHAR(129,"80E0E0E0F0E0C
    ")
780 CALL CHAR(40,"FFFF7EC3C18183
    C3")
790 CALL CHAR(41,"00004070FCFFFF
    FF")
800 CALL CHAR(42,"000000086CFFF
    FF")
810 CALL CHAR(43,"0000000002267F
    FF")
820 CALL CHAR(44,"00000000A0F4FE
```

```
    FF")
830 CALL CHAR(45,"80C0E0FCFDFFFF
    FF")
840 CALL CHAR(46,"01030F1FDFFFFF
    FF")
850 CALL CHAR(47,"FFFFFFFFFFFFFF
    FF")
860 CALL CHAR(130,"")
870 REM  ### WALD PRINTEN
880 FOR J=1 TO 5
890 FOR I=1 TO 32
900 READ ZC
910 CALL HCHAR(J,I,ZC)
920 NEXT I
930 NEXT J
940 S1=15
950 Z=6
960 X=1
970 A=32
980 A=A+1
990 S=S1
1000 FOR J=1 TO X
1010 CALL HCHAR(Z,S,A)
1020 CALL HCHAR(Z,S+2,A)
1030 S=S+2
1040 NEXT J
1050 X=X+1
1060 Z=Z+1
1070 S1=S1-1
1080 ON Z GOTO 990,990,990,990,9
     90,990,990,980,980,990,980,
     990,980,990,990,1090
1090 CALL HCHAR(20,1,39,160)
1100 O=24
1110 FOR P=74 TO 76
1120 CALL HCHAR(22,O,P)
1130 O=O+1
1140 NEXT P
1150 GOTO 1300
1160 REM WIESE BEWEGT
1170 SC=SC-1
1180 FOR ZE=38 TO 39
1190 CALL HCHAR(Z-1,S-1,ZE,3)
1200 FOR ZI=1 TO 50
1210 NEXT ZI
1220 NEXT ZE
1230 T1=1200
1240 FOR TT=1 TO 2
1250 FOR T=T1 TO 1000 STEP -30
1260 CALL SOUND(-100,T,0)
1270 NEXT T
1280 T1=1400
1290 NEXT TT
1300 CALL HCHAR(19,YS,96)
1310 RANDOMIZE
1320 FOR J=1 TO 5
1330 CALL HCHAR(23,5+J,39)
1340 NEXT J
1350 FOR J=1 TO SC
1360 CALL HCHAR(23,5+J,96)
1370 NEXT J
1380 IF SC=0 THEN 2360
1390 CALL HCHAR(2,MS,MG)
1400 MS=MS+1
1410 IF MS<33 THEN 1560
1420 IF F1=1 THEN 1480
1430 F1=1
1440 REM ### TAG/NACHT
1450 MOND=148
1460 GOSUB 2050
1470 GOTO 1560
1480 F1=0
1490 MOND=147
1500 GOSUB 2050
1510 TZAEHL=TZAEHL+1
1520 IF TZAEHL<4 THEN 1560
1530 TZAEHL=0
1540 SC=5
1550 GOTO 1350
1560 CALL GCHAR(2,MS,MG)
1570 IF MG=129 THEN 1610
1580 IF MG<160 THEN 1590 ELSE 16
     00
1590 IF MG>151 THEN 1610 ELSE 16
     00
1600 CALL HCHAR(2,MS,MOND)
1610 PIC=PI(INT(RND*4)+1)
1620 Z=6
1630 S=16
1640 REM ### SPIELBEGINN
1650 CALL HCHAR(6,16,PIC)
1660 CALL HCHAR(6,16,32)
1670 CALL SOUND(-50,-2,7)
1680 CALL HCHAR(Z,S,32)
1690 Z=Z+1
1700 IF Z<15 THEN 1750
1710 IF Z=21 THEN 1160
1720 CALL GCHAR(Z,S,FR)
1730 IF FR=32 THEN 1780
1740 IF FR=96 THEN 1960
1750 W=INT(RND*3)-1
1760 IF W=0 THEN 1750
1770 S=S+W
1780 CALL HCHAR(Z,S,PIC)
1790 CALL KEY(0,K,ST)
1800 IF ST=0 THEN 1870
1810 IF K=68 THEN 1840
1820 IF K=83 THEN 1860
1830 IF K=32 THEN 1680
```

**Listing zu »Garden Hunt« (Fortsetzung)**

# Fallobst für Freaks

```
1840 R=1
1850 GOTO 1870
1860 R=-1
1870 YS=YS+R
1880 IF YS>32 THEN 1890 ELSE 192
     0
1890 YS=YS-R
1900 GOTO 1680
1910 YS=1
1920 CALL HCHAR(19,YS-R,32)
1930 IF YS<1 THEN 1910
1940 CALL HCHAR(19,YS,96)
1950 GOTO 1680
1960 REM
1970 CALL SOUND(-100,-6,0)
1980 CALL SOUND(-100,1200,0)
1990 SCORE=SCORE+1
2000 TE$=STR$(SCORE)&" "
2010 TS=24
2020 TZ=23
2030 GOSUB 2250
2040 GOTO 1380
2050 REM #FARBENSUBROUTINE
2060 REM #TAG UND NACHT
2070 MS=1
2080 CALL COLOR(1,3+F1*10,11+F1*
     -9)
2090 CALL COLOR(2,15+F1*-13,6+F1
     *-1)
2100 IF F1=0 THEN 2130
2110 CALL COLOR(14,16,6+F1*-1)
2120 GOTO 2140
2130 CALL COLOR(14,6+F1*-1,6+F1*
     -1)
2140 CALL COLOR(13,16+F1*-10,6+F
     1*-1)
2150 CALL COLOR(16,16+F1*-10,6+F
     1*-1)
2160 CALL COLOR(5,11+F1*-9,11+F1
     *-9)
2170 CALL COLOR(6,7,11+F1*-9)
2180 CALL COLOR(7,14,11+F1*-9)
2190 CALL COLOR(4,2+F1*14,11+F1*
     -9)
2200 CALL COLOR(8,3,11+F1*-9)
2210 CALL COLOR(3,2+F1*14,11+F1*
     -9)
2220 CALL COLOR(9,2+F1*14,11+F1*
     -9)
2230 CALL COLOR(15,12,6+F1*-1)
2240 RETURN
2250 REM ### ALLGEMEINE PRINTER/
     ROUTINE
2260 J=LEN(TE$)
2270 FOR I=0 TO 2
2280 CALL HCHAR(TZ,TS+I,32)
2290 NEXT I
2300 FOR I=0 TO J
2310 TG=ASC(SEG$(TE$,I+1,1))
2320 CALL HCHAR(TZ,TS+I,TG)
2330 IF TG=32 THEN 2350
2340 NEXT I
2350 RETURN
2360 TZ=22
2370 TS=8
2380 TE$="nochmals_j_n "
2390 GOSUB 2250
2400 CALL KEY(0,K,ST)
2410 IF K=74 THEN 2460
2420 IF K<>78 THEN 2400
2430 CALL CLEAR
2440 PRINT TAB(7);"-AUF WIEDERSE
     HEN-": : : : : : : :
2450 END
2460 CALL HCHAR(22,8,39,14)
2470 SCORE=0
2480 SC=5
2490 GOTO 1300
2500 REM  *** SCHOENSCHRIFT
2510 RESTORE
2520 DATA 003F666C746466EF,007E3
     13B3633337E,003E634140603F1
     E,007E321119113F7E,007F3330
     3C38317F,007F723239383078
2530 DATA 003E6648504F663E,007C6
     03F3132327A03,007C38181C1A1
     83C2,003F13060606263F
2540 DATA 007B72363C363673,00783
     0303072637F
2550 DATA 007F2B2A49494963,007C7
     6323A323477,003E634559617F3
     E,007E33313F3E3078,003E6345
     59617F3E07
2560 DATA 007F313A34323179,003E4
     13C06634F3E
2570 DATA 007F492A0808081C,006F2
     626666E7E37,0073622232321E0
     C,0077222241494936,00793214
     0814264F
2580 DATA 007341623418183C,007F4
     3260C1A317F
2590 FOR J=1 TO 160
2600 READ I
2610 NEXT J
2620 FOR J=65 TO 90
2630 READ Z$
2640 CALL CHAR(J,Z$)
2650 NEXT J
2660 GOSUB 2730
2670 RESTORE
```

```
2680 RETURN
2690 FOR F=1 TO 16
2700 CALL COLOR(F,14,14)
2710 NEXT F
2720 GOTO 2780
2730 CALL SCREEN(14)
2740 FOR J=1 TO 16
2750 CALL COLOR(J,16,14)
2760 NEXT J
2770 RETURN
2780 PRINT "   G A R D E N   H U
     N T"
2790 PRINT "   ----------------
     ----"
2800 PRINT : : :
2810 PRINT "HALLO GAERTNER!": :
2820 PRINT "FANGE MIT DEINEM KOR
     B ": :
2830 PRINT "ALLE FRUECHTE, DIE D
     URCH": :
2840 PRINT "DEN WALD HERUNTERROL
     LEN!": :
2850 PRINT "BENUTZE DAZU DIE TAS
     TEN": :
2860 PRINT "'S' FUER LINKS, 'D'
     FUER": :
2870 PRINT "RECHTS UND 'SPACE',
     UM": :
2880 PRINT "DEINEN KORB ZU STOPP
     EN!": :
2890 PRINT "DRUECK' EINE TASTE!"
     : :
2900 GOSUB 2730
2910 CALL KEY(0,K,ST)
2920 IF ST=0 THEN 2910 ELSE 2930
2930 CALL CLEAR
2940 PRINT TAB(10);"HALT !!": :
     : : : : : : : :
2950 PRINT "FAST HAETT' ICH'S VE
     RGESSEN:": :
2960 PRINT "DU DARFST NUR FUENF
     FEHLER": :
2970 PRINT "MACHEN; NACH ZWEI TA
     GEN": :
2980 PRINT "UND ZWEI NAECHTEN HA
     RTER": :
2990 PRINT "ARBEIT WERDEN DIR AL
     LE FEH-": :
3000 PRINT "LER GESTRICHEN.": :
3010 PRINT "DRUECK' EINE TASTE!"
3020 CALL KEY(0,K,ST)
3030 IF ST=0 THEN 3020
3040 CALL CLEAR
3050 RETURN
```

**Listing zu »Garden Hunt« (Schluß)**

# Rauhe Sitten - und was wir davon halten

An dieser Stelle sollte ursprünglich ein anderes Programm als Listing des Monats prämiert werden. Alles war fix und fertig vorbereitet. Da rief ein aufmerksamer Leser in der Redaktion an und wies darauf hin, daß ihm das Bildschirmfoto aus der Vorschau in Ausgabe 5 sehr bekannt vorkäme. Er habe das Programm dazu bereits vor einigen Monaten in einer anderen Computerzeitschrift gesehen und daraus abgetippt. Wir überprüften die Sache. Die beiden Programme waren nicht identisch, aber die Grafik und — dies ist der entscheidende Aspekt — die Spielidee glichen sich nahezu völlig.

Nun betrachten wir das Listing des Monats aber als etwas besonderes. Die Höhe der Prämie ist keine schlichte Aufrechnung und Honorierung der Programmierarbeit, daher richtet sie sich auch nicht nach dem tatsächlichen Umfang eines Programms (so scheinen allerdings eine Reihe unserer Leser die 2000-Mark-Prämie fälschlicherweise zu interpretieren). Für den gleichen Betrag müßte ein normal Berufstätiger ohne Fachqualifikation immerhin fast einen Monat lang ganztags arbeiten gehen!

Dem Wettbewerb liegt vielmehr die Absicht zugrunde, die Kreativität unter den Hobbyprogrammierern fördern zu wollen. Vor allem auf die Pfiffigkeit und Originalität der Idee kommt es uns an; saubere Umsetzung und Ausführung ist selbstverständliche Voraussetzung. Die Höhe der Prämie soll Anreiz zur Teilnahme am »Wettkampf der Köpfe« für alle Leser und materielle Förderung für die Weiterentwicklung der Prämierten sein, kein »Entgelt« für geopferte Freizeit, denn wer darin ein »Opfer« sieht, sollte sich überlegen, ob er überhaupt noch einem »Hobby« nachgeht.

Etwas stimmt uns an der Sache besonders traurig: In den Zeiten der mühevoll selbst zusammengelöteten Kleincomputer gehörte es einmal zum guten Ton unter den Computerfreaks, daß in Programmen mit fremden Programmideen wenigstens der Name des eigentlichen Urhebers genannt wurde. Dabei vergab sich niemand etwas. Heute scheint es mehr und mehr zum guten (?) Ton zu gehören, möglichst geschickt zu kopieren und möglichst schnell »abzusahnen«.

Dabei sind sich viele nicht darüber im Klaren, wie schnell, besonders im kommerziellen Bereich, die Grenze vom scheinbar harmlosen Kavaliersdelikt zur kriminellen Handlung überschritten ist. Die Software-Händler wissen ein Lied davon zu singen. Wir werden in einer der nächsten Ausgaben ausführlich zu diesem Thema Stellung beziehen. (lg)

**Haben Sie Schwierigkeiten mit dem Lernen von Vokabeln? Verwenden Sie doch einfach Ihren TI 99/4A mit dem Extended Basic Modul und das nachfolgende Programm, um Ihre Vokabeln zu pauken.**

# VOKABELTRAINER

Das Programm »Vokabeltrainer« bietet die Möglichkeit, Vokabeln einzugeben und bestimmten Oberbegriffen zuzuordnen, die auch auf Kassette gespeichert werden können, um später erneut verwendet werden zu können. Folgende Funktionen stehen zur Verfügung:

☐ Die eingegebenen Vokabeln können aufgelistet und, wenn nötig, korrigiert werden.

☐ Das Unterprogramm »Abfragen« bietet die Möglichkeit, entweder einzelne Begriffe abzurufen, ähnlich einem Lexikon, oder durch Eingabe eines Oberbegriffes ganze Gruppen von Vokablen zu erfragen (vergleichbar mit einer Kartei).

☐ Die dritte Funktion ermöglicht es, sich vom Computer aus einem vorher festgelegten Bereich abfragen zu lassen. Die Anzahl der zur Verfügung stehenden Versuche muß vom Benutzer bestimmt werden.

Nach Programmstart wird man aufgefordert, eine Funktion zu wählen. Drückt man die Taste »1«, so erscheint ein Begleittext und im untersten Feld des Bildschirms erwartet der Computer eine Eingabe (erste Vokabel). Dann wird die zweite Vokabel erfragt, und so weiter. Will man das Unterprogramm verlassen, muß man bei der Abfrage von Begriff 1 »ENTER« eingeben. Vor Abschluß des Unterprogramms besteht die Möglichkeit, die vorher eingegebene Vokabelpaare mit einem Oberbegriff zu versehen.

In den Unterprogrammen 2 und 3 erfragt der Computer zunächst die nötigen Informationen. Danach erfolgt im unteren Bildschirmteil eine Eingabe. So lange man nicht »ENTER« eingibt, bleibt man im gewählten Programmteil. Andernfalls verläßt man, analog zu Unterprogramm 1, die Funktion und kehrt zum Hauptmenü zurück. Die gleiche Art der Eingabe kann man auch im Unterprogramm 6 (Korrigieren) anwenden. Nur bezieht sie sich dort auf die Funktionen »Ändern«, »nächste Folge« (für den Fall, daß nicht alle eingegebenen Begriffspaare auf einmal gelistet werden können weil der Bildschirmbereich zu klein ist) und »Rücksprung«. Diese Funktionen können durch Eingabe der entsprechenden Kürzel ausgewählt werden.

## Anwendungsbeispiele

1. Beispiel: Englischvokabeln
Begriff 1: To have
Begriff 2: Haben ... und so weiter
Oberbegriff: »Englisch Lektion 1« (oder: »Englisch: Text, Autor«)

2. Beispiel: Geschichtsdaten
Begriff 1: 1789
Begriff 2: Französiche Revolution ... und so weiter
Oberbegriff: »Geschichte: 1700 bis 1800«

3. Beispiel: Schallplattenarchiv
Begriff 1: Titel
Begriff 2: Sänger
Begriff 3: Lieder
Begriff 4: Seite 1
Begriff 5: Titel 1 ... und so weiter
Oberbegriff: »Titel und Sänger«

Zu Beispiel 1: Man kann die Vokabeln nach Lektionen geordnet zusammenfas-

```
10 REM VOKABELTRAINER FUER
     TI 99/4A + EXTENDED
   BASIC
20 REM (C) JOACHIM THOMAS
        MOSELSTR.3
      5592 KLOTTEN
100 CALL CLEAR
110 CALL CHAR(95,"000000FFFF0000
    00")
120 OPTION BASE1
130 DIM A$(400),B$(400),OBG$(50)
    ,OBG(50)
140 CALL DIS
150 DISPLAY AT(2,7):"VOKABELTRAI
    NER" :: DISPLAY AT(4,2):"Bit
    te eine Funktion waehlen"
160 CALL KEY(0,K,S):: IF S=0 THE
    N 160 :: IF K<49 XOR K>54 TH
    EN 160
170 ON K-48 GOSUB 1000,2000,3000
    ,4000,5000,6000
180 CALL DELSPRITE(#1):: GOTO 14
    0
1000 !----------------------
1010 CALL SPRITE(#1,62,2,41,8)::
     CALL CLEAN(1,4,1,1)
1020 DISPLAY AT(1,1):"Nach Einga
    be eines Begriffs-" :: DISP
    LAY AT(2,1):"paares:"
1030 DISPLAY AT(3,1):"Weitere Ei
    ngabe=Weiter" :: DISPLAY AT
    (4,9):"'Enter'=Ruecksprung"
1040 Z=Z+1
1050 ACCEPT AT(20,6)BEEP:A$(Z)::
     IF A$(Z)="" THEN 1080
1060 ACCEPT AT(22,6)BEEP:B$(Z)
1070 GOTO 1040
1080 CALL CLEAN(1,4,1,1):: DISPL
    AY AT(1,7):"Oberbegriff ?"
    :: DISPLAY AT(2,1):"Oberbeg
    riff oder 'Enter' einge-"
1090 DISPLAY AT(3,1):"ben:" :: A
    CCEPT AT(4,1)BEEP:C$ :: IF
    C$="" THEN 1110
1100 U=U+1 :: OBG$(U)=C$ :: OBG(
    U)=Z-1
1110 Z=Z-1 :: RETURN
2000 !----------------------
2010 CALL SPRITE(#1,62,2,57,8)::
     CALL CLEAN(1,4,1,1)
2020 DISPLAY AT(1,1):"Abfrage na
    ch Oberbegriffen ?" :: DISP
    LAY AT(3,4):"'J' oder 'N' d
    ruecken"
2030 CALL KEY(0,K,S):: IF S=0 TH
    EN 2030 :: IF K<>78 AND K<>
    74 THEN 2030
2040 IF K<>78 THEN 2250
2050 DISPLAY AT(1,1):"Sprache 1
      ->  Sprache 2  ?"
2060 CALL KEY(0,K,S):: IF S=0 TH
    EN 2060 :: IF K<>78 AND K<>
    74 THEN 2060
```

```
2070 IF K<>78 THEN W=0 ELSE W=1
2080 DISPLAY AT(1,1):"Nach Einga
     be und Beantwor-" :: DISPLA
     Y AT(2,1):"tung:"
2090 DISPLAY AT(3,1):"Weitere Ei
     ngabe=Weiter" :: DISPLAY AT
     (4,9):"'Enter'=Ruecksprung"
2100 IF W=1 THEN 2180
2110 ACCEPT AT(20,6)BEEP:C$ :: I
     F C$="" THEN 2380
2120 FOR I=1 TO Z
2130 IF C$<>A$(I)THEN 2160
2140 DISPLAY AT(22,6):B$(I)
2150 CALL CLEAN(20,24,2,8):: GOT
     O 2110
2160 NEXT I
2170 DISPLAY AT(22,6):"Nicht vor
     handen !" :: CALL CLEAN(20,
     24,2,8):: GOTO 2110
2180 ACCEPT AT(22,6)BEEP:C$ :: I
     F C$="" THEN 2380
2190 FOR I=1 TO Z
2200 IF C$<>B$(I)THEN 2230
2210 DISPLAY AT(20,6):A$(I)
2220 CALL CLEAN(20,24,2,8):: GOT
     O 2180
2230 NEXT I
2240 DISPLAY AT(20,6):"Nicht vor
     handen !" :: CALL CLEAN(20,
     24,2,8):: GOTO 2180
2250 CALL DELSPRITE(#1):: CALL C
     LEAN(1,4,1,1):: CALL CLEAN(
     6,17,1,1)
2260 DISPLAY AT(1,2):"Bitte Ober
     begriff eingeben" :: ACCEPT
      AT(2,2)BEEP:C$ :: IF C$=""
      THEN 2380
2270 FOR I=1 TO U :: IF C$<>OBG$
     (I)THEN 2290
2280 IF I=1 THEN R=1 :: E=OBG(I)
     :: J=10 :: GOTO 2300 ELSE R
     =OBG(I-1)+1 :: E=OBG(I):: J
     =10 :: GOTO 2300
2290 NEXT I :: DISPLAY AT(2,2):"
     Nicht vorhanden !" :: CALL
     DELAY :: GOTO 2260
2300 P=R-1 :: FOR I=R TO E
2310 DISPLAY AT(I-P+5,1):I;" ";A
     $(I);" ";B$(I):: IF I=R+J T
     HEN 2330 :: IF LEN(A$(I))+L
     EN(B$(I))>20 THEN P=P-1 ::
     J=J-1
2320 NEXT I :: GOTO 2360
2330 DISPLAY AT(3,2):"Weiter ?"
     :: ACCEPT AT(3,10)BEEP:C$
2340 IF C$<>"J" THEN 2360
2350 R=I+1 :: J=10 :: CALL CLEAN
     (6,18,1,1):: GOTO 2300
2360 DISPLAY AT(3,1):"Beliebige
     taste druecken"
2370 CALL KEY(0,K,S):: IF S=0 TH
     EN 2370 :: CALL CLEAN(1,4,1
     ,1):: CALL CLEAN(6,18,1,1):
     : GOTO 2260
2380 RETURN
3000 !----------------------
3010 CALL SPRITE(#1,62,2,73,8)::
     CALL CLEAN(1,4,1,1)
3020 DISPLAY AT(1,1):"Welcher Be
     reich ?" :: DISPLAY AT(2,1)
     :"Oberbegriff oder 'ALLES'
     einge-"
3030 DISPLAY AT(3,1):"ben:" :: A
     CCEPT AT(3,5)BEEP:C$ :: IF
     C$="ALLES" THEN 3070
3040 FOR I=1 TO U :: IF C$<>OBG$
     (I)THEN 3060
3050 IF I=1 THEN R=1 :: E=OBG(I)
     :: GOTO 3080 ELSE R=OBG(I-1
     )+1 :: E=OBG(I):: GOTO 3080
3060 NEXT I :: DISPLAY AT(3,5):"
     Nicht vorhanden !" :: CALL
     DELAY :: GOTO 3030
3070 R=1 :: E=Z
3080 CALL CLEAN(1,4,1,1):: DISPL
     AY AT(1,2):"Zahl der Versuc
     he ?" :: ACCEPT AT(1,23)BEE
     P:V
3090 DISPLAY AT(1,2):"Sprache 1
      -> Sprache 2 ?" :: DISPL
     AY AT(3,2):"'J' oder 'N' dr
     uecken"
3100 CALL KEY(0,K,S):: IF S=0 TH
     EN 3100 :: IF K<>78 AND K<>
     74 THEN 3100
3110 IF K<>78 THEN W=0 ELSE W=1
3120 DISPLAY AT(1,1):"Nach Vorga
     be eines begriffs:" :: DISP
     LAY AT(3,2):"Beantwortung=W
     eiter"
3130 DISPLAY AT(4,7):"'Enter'=Ru
     ecksprung" :: IF W=1 THEN 3
     190
3140 RANDOMIZE :: S=INT((E-R+1)*
     RND)+R :: Q=V
3150 DISPLAY AT(20,6):A$(S):: AC
     CEPT AT(22,6)BEEP:C$
3160 IF C$="" THEN 3240 :: IF C$
     <>B$(S)THEN Q=Q-1 :: GOTO 3
     180
3170 DISPLAY AT(24,6):"Richtig !
     " :: CALL CLEAN(20,24,2,8):
     : GOTO 3140
3180 DISPLAY AT(24,6):"Falsch !"
      :: CALL CLEAN(20,24,2,8)::
      IF Q>0 THEN 3150 ELSE 3140
3190 RANDOMIZE :: S=INT((E-R+1)*
     RND)+R :: Q=V
3200 DISPLAY AT(22,6):B$(S):: AC
     CEPT AT(20,6)BEEP:C$
3210 IF C$="" THEN 3240 :: IF C$
     <>A$(S)THEN Q=Q-1 :: GOTO 3
     230
3220 DISPLAY AT(24,6):"Richtig !
     " :: CALL CLEAN(20,24,2,8):
     : GOTO 3190
3230 DISPLAY AT(24,6):"Falsch !"
      :: CALL CLEAN(20,24,2,8)::
      IF Q>0 THEN 3200 ELSE 3190
3240 RETURN
4000 !----------------------
4010 CALL CLEAR
4020 OPEN #1:"CS1",SEQUENTIAL ,I
     NTERNAL ,OUTPUT ,FIXED
```

Listing zum Programm »Vokabeltrainer«

```
4030 PRINT #1:Z,U :: CLOSE #1 ::
     OPEN #2:"CS1",SEQUENTIAL ,
     INTERNAL ,OUTPUT ,FIXED 128
     :: FOR I=1 TO U STEP 7
4040 PRINT #2:OBG$(I),OBG(I),OBG
     $(I+1),OBG(I+1),OBG$(I+2),O
     BG(I+2),OBG$(I+3),OBG(I+3),
     OBG$(I+4),OBG(I+4),OBG$(I+5
     ),OBG(I+5),OBG$(I+6),OBG(I+
     6)
4050 NEXT I :: CLOSE #2
4060 OPEN #3:"CS1",SEQUENTIAL ,I
     NTERNAL ,OUTPUT ,FIXED 170
4070 FOR I=1 TO Z STEP 8
4080 PRINT #3:A$(I),B$(I),A$(I+1
     ),B$(I+1),A$(I+2),B$(I+2),A
     $(I+3),B$(I+3),A$(I+4),B$(I
     +4),A$(I+5),B$(I+5),A$(I+6)
     ,B$(I+6),A$(I+7),B$(I+7)
4090 NEXT I :: CLOSE #3
4100 RETURN
5000 !---------------------
5010 CALL CLEAR :: OPEN #1:"CS1"
     ,SEQUENTIAL ,INTERNAL ,INPU
     T ,FIXED
5020 INPUT #1:Z,U :: CLOSE #1 ::
     OPEN #2:"CS1",SEQUENTIAL ,
     INTERNAL ,INPUT ,FIXED 128
     :: FOR I=1 TO U STEP 7
5030 INPUT #2:OBG$(I),OBG(I),OBG
     $(I+1),OBG(I+1),OBG$(I+2),O
     BG(I+2),OBG$(I+3),OBG(I+3),
     OBG$(I+4),OBG(I+4),OBG$(I+5
     ),OBG(I+5),OBG$(I+6),OBG(I+
     6)
5040 NEXT I :: CLOSE #2
5050 OPEN #3:"CS1",SEQUENTIAL ,I
     NTERNAL ,INPUT ,FIXED 170
5060 FOR I=1 TO Z STEP 8
5070 INPUT #3:A$(I),B$(I),A$(I+1
     ),B$(I+1),A$(I+2),B$(I+2),A
     $(I+3),B$(I+3),A$(I+4),B$(I
     +4),A$(I+5),B$(I+5),A$(I+6)
     ,B$(I+6),A$(I+7),B$(I+7)
5080 NEXT I
5090 CLOSE #3
5100 RETURN
6000 !---------------------
6010 CALL CLEAN(1,4,1,1):: CALL
     CLEAN(6,17,1,1)
6020 DISPLAY AT(2,1):"Korrigiere
     n von Oberbegriffen" :: DIS
     PLAY AT(4,1):"'J' oder 'N'
     druecken"
6030 CALL KEY(0,K,S):: IF S=0 TH
     EN 6030 :: IF K<>78 AND K<>
     74 THEN 6030
6040 IF K<>78 THEN 6160
6050 R=1 :: P=0 :: J=10
6060 FOR I=R TO Z
6070 DISPLAY AT(I-P+5,1):I;" ";A
     $(I);" ";B$(I):: IF I=R+J T
     HEN 6090 :: IF LEN(A$(I))+L
     EN(B$(I))>20 THEN P=P-1 ::
     J=J-1
6080 NEXT I
6090 DISPLAY AT(1,1):"A  fuer ae
     ndern" :: DISPLAY AT(2,1):"
     N  fuer naechste Folge"
```

```
6100 DISPLAY AT(3,1):"R  fuer Ru
     ecksprung" :: ACCEPT AT(4,1
     )BEEP:C$
6110 IF C$="A" THEN 6130 ELSE IF
      C$="N" THEN 6120 ELSE RETU
     RN
6120 J=10 :: R=I+1 :: P=R-1 ::'C
     ALL CLEAN(6,18,1,1):: GOTO
     6060
6130 DISPLAY AT(4,1):"Welche Nr.
      ?" :: ACCEPT AT(4,12)BEEP:
     H
6140 ACCEPT AT(20,6)BEEP:A$(H)::
      ACCEPT AT(22,6)BEEP:B$(H)
6150 CALL CLEAN(20,24,2,8):: GOT
     O 6090
6160 R=1 :: P=0
6170 FOR I=R TO U
6180 DISPLAY AT(I-P+5,1):I;" ";O
     BG$(I)
6190 IF I=R+10 THEN 6210
6200 NEXT I
6210 DISPLAY AT(1,1):"A  fuer ae
     ndern" :: DISPLAY AT(2,1):"
     N  fuer naechste Folge"
6220 DISPLAY AT(3,1):"R  fuer Ru
     ecksprung" :: ACCEPT AT(4,1
     )BEEP:C$
6230 IF C$="A" THEN 6250 ELSE IF
      C$="N" THEN 6240 ELSE RETU
     RN
6240 R=I+1 :: P=R-1 :: CALL CLEA
     N(6,17,1,1):: GOTO 6170
6250 DISPLAY AT(4,1):"Welche nr.
      ?" :: ACCEPT AT(4,12)BEEP:
     H
6260 ACCEPT AT(4,1):OBG$(H):: GO
     TO 6210
6270 RETURN
8000 SUB DIS
8010 CALL CLEAN(6,18,1,1):: CALL
      CLEAN(20,24,2,8):: CALL CL
     EAN(1,4,1,1):: CALL CLEAN(2
     1,23,2,1)
8020 DISPLAY AT(6,2):"1    EINGE
     BEN" :: DISPLAY AT(8,2):"2
       ABFRAGEN"
8030 DISPLAY AT(10,2):"3    ABGE
     FRAGT WERDEN" :: DISPLAY AT
     (12,2):"4    ABSPEICHERN"
8040 DISPLAY AT(14,2):"5    EINL
     ESEN" :: DISPLAY AT(16,2):"
     6    KORRIGIEREN"
8050 DISPLAY AT(20,1):"B. 1:" ::
      DISPLAY AT(22,1):"B. 2:"
8060 DISPLAY AT(24,1):"LOE.:" ::
      CALL HCHAR(5,1,95,32):: CA
     LL HCHAR(19,1,95,32)
8070 SUBEND
8080 SUB CLEAN(A,B,C,D)
8090 IF A=20 THEN CALL DELAY
8100 FOR I=A TO B STEP C
8110 CALL HCHAR(I,D,32,31-D)
8120 NEXT I
8130 SUBEND
8140 SUB DELAY
8150 FOR I=1 TO 500 :: NEXT I
8160 SUBEND
```

Listing zum Programm »Vokabeltrainer« (Schluß)

sen oder die zu einem Text gehörenden Vokabeln mit dem Titel des gelesenen Stücks versehen.

Zu Beispiel 2: Die Geschichtsdaten können nach Zeiträumen oder Epochen überschrieben werden, so daß man parallel zum Geschichtsbuch (mit der gleichen Zeiteinteilung) arbeiten kann. So weiß man immer, welchen Oberbegriff man wählen muß, nämlich die Überschrift des entsprechenden Kapitels im Buch.

Zu Beispiel 3: Angaben über Schallplatten können unter dem Titel der Platte gespeichert werden und durch Eingabe dieses Titels auch wieder abgerufen werden. Es ist sinnvoll, den Plattentitel und den Interpreten auch als »Vokabel« anzugeben, da der Oberbegriff nicht mit den Begriffspaaren zusammen aufgelistet wird.

Man kann den »Vokabeltrainer« auch, analog zu Beispiel 3, zur Archivierung anderer Informationen verwenden: Bücher, Briefmarken oder auch persönliche Daten von Clubmitgliedern können dann abgespeichert werden, wobei der Oberbegriff als Passwort dienen kann.

(Joachim Thomas)

| | |
|---|---|
| Zeile 100 bis 150 | Bildschirmaufbau (mittels SUB DIS) und Variablendimensionierung) |
| Zeile 160 bis 170 | Abfrage des gewünschten Unterprogramms und entsprechende Verzweigung |
| **Unterprogramm eingeben** | |
| Zeile 1000 bis 1030 | Plazierung des Programmzeigers und Ausgabe des erklärenden Begleittextes |
| Zeile 1040 bis 1070 | Eingabe der Begriffspaare (Ausstieg mit »ENTER«) |
| Zeile 1080 bis 1100 | Wenn erwünscht, Eingabe eines Oberbegriffes und Zuordnung der Obergrenze |
| Zeile 1110 | Löschen der zuletzt eingegebenen Vokabel (" ") und Rücksprung in Zeile 180 (Überleitung zum Hauptmenü) |
| **Unterprogramm »Abfragen«** | |
| Zeile 2000 bis 2100 | Festlegung des Arbeitsmodus, Plazierung des Begleittextes und entsprechende Verzweigung |
| Zeile 2110 bis 2170 | Eingabe von Begriff 1 und Suche von Begriff 2; dann Ausgabe des Begriffs beziehungsweise Fehlermeldung |
| Zeile 2180 bis 2240 | Gleicher Vorgang wie vorher, nur Eingabe von Begriff 2 und Suche von Begriff 1 |
| Zeile 2250 bis 2290 | Eingabe des Oberbegriffs, dessen zugehörige Vokabeln gelistet werden sollen; Festlegung der Variablen zur Bildschirmausgabe |
| Zeile 2300 bis 2380 | Auflisten der Begriffspaare; nach vollständer Durchführung Rücksprung in das Hauptmenü |
| **Unterprogramm »Abgefragt werden«** | |
| Zeile 3000 bis 3130 | Festlegung des Arbeitsmodus, Plazierung des Begleittextes und entsprechende Verzweigung |
| Zeile 3140 bis 3190 | Ermittlung und Vorgabe von Begriff 1 und Abfrage des Gegenbegriffes; Auswertung des Ergebnisses und entsprechende Ausgabe (Richtig oder Falsch), verbunden mit Korrektur der noch möglichen Versuche (Variable V) |
| Zeile 3190 bis 3230 | Gleicher Vorgang, nur mit Vorgabe von Begriff 2 |
| Zeile 3240 | Rücksprung zum Hauptmenü |
| **Unterprogramm »Abspeichern«** | |
| Zeile 4000 bis 4100 | Abspeichern der Vokabeln und Oberbegriffe auf Kassette (durch Verringerung der Schleifendurchgänge (STEP 7/STEP 8) beschleunigt) |
| **Unterprogramm »Einlesen«** | |
| Zeile 5000 bis 5100 | Wie Unterprogramm »Abspeichern«, beschleunigter Einlesevorgang |
| **Unterprogramm »Korrigieren«** | |
| Zeile 6000 bis 6050 | Abfrage des Arbeitsmodus und Verzweigung |
| Zeile 6060 bis 6080 | Belegung der benötigten Variablen und Ausgabe auf dem Bildschirm |
| Zeile 6090 bis 6150 | Abfrage und Durchführung der gewünschten Funktion (Ausstieg mit R für Rücksprung) |
| Zeile 6160 bis 6200 | Ausgabe der Oberbegriffe auf dem Bildschirm |
| Zeile 6210 bis 6260 | Abfrage und Durchführung der gewünschten Funktion |
| Zeile 6270 | Rücksprung zum Hauptmenü |

Sub Dis
Bildschirmaufbau
Sub Clean
Löschen von Bildschirmbereichen in Abhängigkeit von A,B,C,D
Sub Delay
Verzögerung des Löschvorgangs nach erfolgter Bildschirmausgabe

**Programmbeschreibung**

| | |
|---|---|
| A$(400) | Feld für den ersten Begriff eines Vokabelpaares |
| B$(400) | Feld für den entsprechenden Gegenbegriff |
| OBG$(50) | Feld für Oberbegriffe |
| OBG(50) | Obergrenzen der von den Oberbegriffen abgedeckten Bereiche |
| Z | Anzahl der bisher eingegebenen/eingelesenen Begriffspaare |
| U | Anzahl der bisher verwendeten Oberbegriffe |
| A,B,C,D | Angaben über den zu löschenden Bildschirmbereich |
| C$ | Allgemeine Eingabevariable für Strings |
| I | Allgemeine Laufvariable |
| R | Index des ersten zu verwendenden Begriffspaares |
| E | Index des letzten zu verwendenden Begriffspaares |
| P | Richtige Ausgabestelle auf dem Bildschirm |
| S | Zufallszahl zur Ermittlung einer vorzugebenden Vokabel |
| W | Verzweigungsvariable |
| J | Anzahl der zu listenden Vokalpaare |
| H | Index des zu berichtigenden Vokalpaares |
| V | Anzahl der maximalen Versuche |
| Q | Anzahl der noch möglichen Versuche |

**Variablenliste**

# Kassetten-Dire

**Das Programm erleichtert das und Auffinden von Programm Commodore 64. Es wird am Seite einer Kassette gespeich der auf der Kassette gespeic in Form einer Zeitkonstante, das gesuchte Programm aufgefunden werden kann.**

```
                              *********************************
                              *
2 rem **********************************
4 rem * cassette directory
6 rem **********************************
8 :
10 rem/pro-510-copyright g.prochaska 1983/
20 print chr$(14):poke53280,12:poke53281,15:print"":print:p
30 print"*--------------------------------------*";
40 print"* ***** menue-cassette xx/side y *****";
50 print"*--------------------------------------*":print
60 fori=1to8:read na$(i),z$(i):next
70 fori=1to8
80 print"      ";i;"*";na$(i);tab(27);"*"
90 next:print:print
95 print"*--------------------------------------*";
99 :
100 print"* *****     bitte Zahl eingeben   *****";
105 print"*--------------------------------------*":print
110 geta$:ifval(a$)<1orval(a$)>8then110
115 a=val(a$)
130 print"";
135 l$="|        |        |        |        |"
140 print l$;
150 print"|RECORD |  PLAY |  REWIND |  F.FWD |STOP |";
160 printl$;
170 a1$="|        |        |        |press|        |"
180 a2$="|        |        |press|        |        |"
190 a3$="|        |press|        |        |        |"
210 if(peek(1)and16)=0thenprinta1$;:goto220
215 printl$:goto230
220 printl$:wait1,24
230 ifa=1thenprint"";a3$:goto280
240 print "";a2$;l$;
250 if(peek(1)and16)<>0then250
255 print"";l$;
260 ti$="000000"
270 ifti$<>z$(a)then270
280 print"";a1$;l$
290 sys 64659:if(peek(1)and16)=0then290
300 print"load";chr$(34);na$(a);chr$(34)
310 print"  << press return >>  ":end
1000 data"programm 1                          "
1005 data 000000
1010 data"programm 2                          "
1015 data 000011
1020 data"programm 3                          "
1025 data 000022
1030 data"programm 4                          "
1035 data 000033
1040 data"programm 5                          "
1045 data 000044
1050 data"programm 6                          "
1055 data 000055
1060 data"programm 7                          "
1065 data 000106
1070 data"programm 8                          "
1075 data 000117
ready.
```

Programm zur Erstellung eines Kassetten-Directories

```
= reverse on
= reverse off
= cursor home
= clear screen
= cursor down
= cursor up
= Zeichenfarbe rot
= Zeichenfarbe blau
= Zeichenfarbe orange
```

*Die Steuerzeichen*

# Atari-Joys

**Mancher Spielefreak möchte auch am TI 99/4A gerne bessere Joysticks als die originalen von Texas Instruments anschließen — ganz abgesehen davon, daß mit Auslaufen des TI-Heimcomputers auch die zugehörigen Joysticks im Handel rar werden. Wenn man doch nur die mit dem Atari-Anschluß verwenden könnte.**

Buchse:

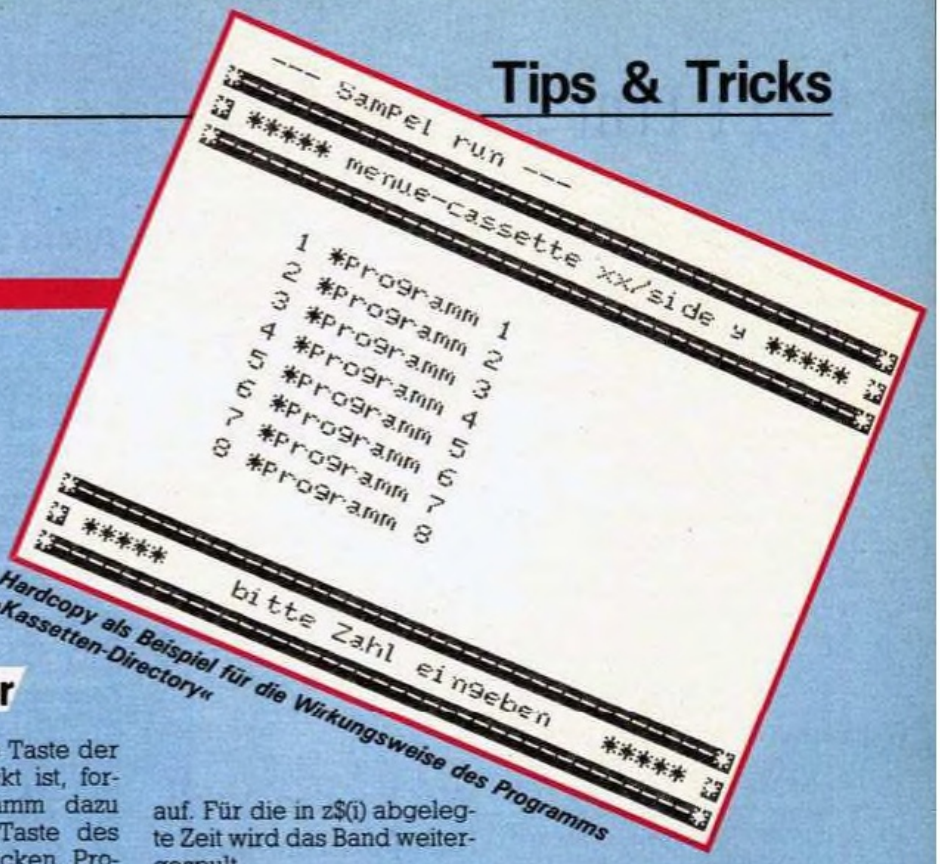
3 links
4 rechts
5 ohne Anschluß (manchmal Paddle-Eingang)
6 Feuerknopf (bei manchen kompatiblen Eingang für Lichtgriffel)
7 +5 Volt/100 mA
8 Masse
9 ohne Anschluß (manchmal Paddle-Eingang)

Kontakte:
1 aufwärts
2 abwärts

**Bild 1. Belegung der Atari-Joystick-Buchse**

# ctory

## Abspeichern
## en auf der Datasette des
## Anfang einer jeden
## ert und enthält die Namen
## nerten Programme sowie,
## die Stelle der Kassette, an der




**Hardcopy als Beispiel für die Wirkungsweise des Programms »Kassetten-Directory«**

Im folgenden wird der Programmablauf beschrieben.

1. Nach »RUN« gibt das Programm eine Liste der auf der Kassette vorhandenen Programme aus und fordert zur Eingabe einer Zahl (entsprechend dem gewünschten Programm) auf.
2. Falls noch eine Taste der Datasette gedrückt ist, fordert das Programm dazu auf, die »STOP«-Taste des Recorders zu drücken. Programmfortsetzung erfolgt erst nach Drücken der »STOP«-Taste.
3. Falls nicht das erste Programm auf der Kassette geladen werden soll, fordert das Programm jetzt zum Drücken der »F.FWD«-Taste auf. Für die in z$(i) abgelegte Zeit wird das Band weitergespult.
4. Wenn das Band die vorgegebene Stelle erreicht hat, wird der Kassettenmotor mit »SYS 64659« abgeschaltet und der Benutzer zum Drücken der »STOP«-Taste aufgefordert.
5. Auf dem Bildschirm erscheint jetzt: load"programmname"

Nach Drücken von »RETURN« und »PLAY« wird das Programm geladen. Soll ein Programm an dieser Stelle gespeichert werden, so ist die Kassette zu entfernen, das Programm wird von einer anderen Kassette geladen, die erste Kassette wird wieder eingelegt und das Programm an der vorgegebenen Stelle gespeichert.

(Gunter Prochaska)

# tick für den TI 99/4A

Man kann. Dazu braucht man nur einen kleinen Adapter. Für zirka zehn Mark Material und eine Stunde Arbeit ist so ein Übergangsstück leicht herzustellen. Sie benötigen dazu:

einen Lötkolben (15 bis 30 Watt)
Lötzinn
1 AMP-Trapez-Stecker, 9polig, weiblich
2 AMP-Trapez-Buchsen, 9polig, männlich
2 Widerstände 330 Ohm, je ¼ Watt
8 Siliziumdioden (zum Beispiel 1N4148)
ein paar Schrauben
ein Gehäuse
ein Stück 7adriges Kabel.

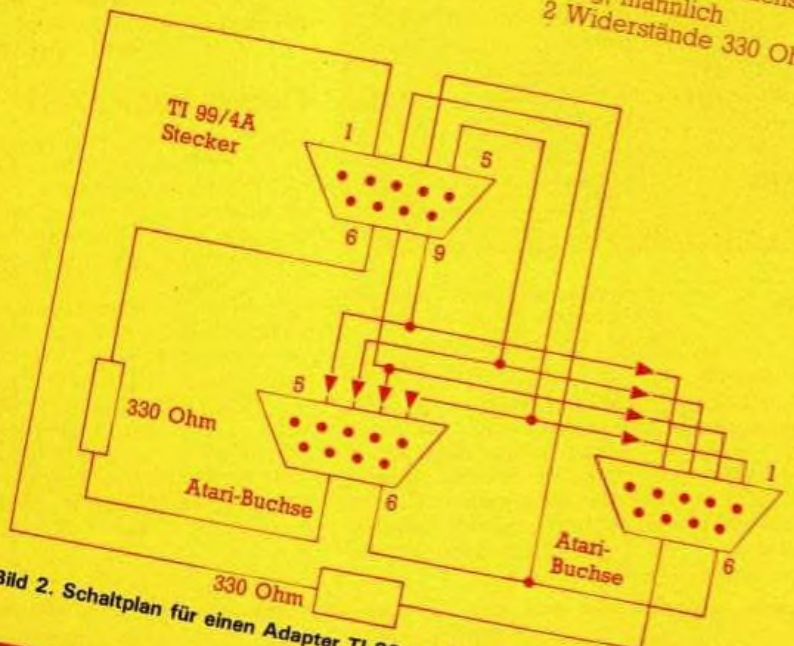



**Bild 2. Schaltplan für einen Adapter TI 99/Atari-Joystick**

Betrachten wir uns zuerst die Anschlußbelegung der Atari-Joysticks in Bild 1 (Gilt natürlich auch für kompatible Fabrikate).

Die Belegung des TI-Joystickanschlusses können Sie der Ausgabe 5 von Happy-Computer entnehmen. Für den Adapter zwischen TI 99/4A und Atari-Joysticks ergibt sich ein Schaltplan wie in Bild 2. Wer nur mit einem Joystick spielen will, kann die rechte untere Buchse (Joystick 2) zusammen mit den zugehörigen vier Dioden weglassen.

(Guido Stumpel/lg)

# Damast ~ ein guter Stoff

**Allen TI 99/4A-Besitzern zum Trost: Obwohl Texas Instruments die Produktion des Computers eingestellt hat, bieten zahreiche Firmen neue Hard- und Software an. Ein neues Software-Paket trägt den Namen Damast und verspricht Datenbankfunktionen. Damast steht für Datenmanagementsystem und ist für die Verwaltung mittelgroßer Datenmengen auf dem TI 99/4A gedacht.**

Unter mittelgroßer Datenmenge sind — laut Autor — zirka 300 bis 400 elektronische »Karteikarten« (bei Mindestgerätekonfiguration) zu verstehen. Das Programm ist in dieser Beziehung also, wie der Hersteller in seinem Prospekt auch kundtut, zum Beispiel zur Mitgliederverwaltung von Vereinen, Kundenverwaltung von kleineren Betrieben und Klientenverwaltung von Maklern geeignet.

Bei Damast handelt es sich übrigens nicht, wie man vielleicht denken könnte, um ein einzelnes Programm, sondern um eine ganze Reihe von Programmodulen (das heißt Programmteile, die in sich geschlossen und getrennt auf Diskette gespeichert sind). Sie sind einzeln erhältlich und ergänzen sich alle zusammen zum kompletten Datenverarbeitungssystem. Je nach den gestellten Anforderungen kann man sich so sein individuelles System zusammenstellen und jederzeit nach Belieben erweitern.

Der Kern des ganzen Systems besteht aus dem Kartenverwaltungs- und Analysemodul. Mit diesem Teilsystem ist es dem Anwender schon möglich, Karteikartendaten zu bearbeiten, das heißt den Inhalt anzuzeigen, Eintragungen, Änderungen und Löschungen vorzunehmen.

Zur besseren Übersicht und zur Analyse der gesammelten Daten ist außerdem ein Unterprogramm integriert, welches für die Anzeige der Grunddaten, des Inhalts der Datenbank und des Speicherbelegungsgrades in Prozent auf Drucker oder Bildschirm zuständig ist. Mit Hilfe eines Druckprogramms und des Drucklistengenerators kann man die Daten, die man in den Karteikarten gespeichert hat, in beliebiger Form zu Papier bringen. Auf diese Weise können zum Beispiel Adreßetiketten oder Formulare beschriftet werden. Hierbei kann man bis zu drei Drucker ansteuern.

Ebenfalls ausdrucken lassen sich Serienbriefe, die man mit dem Texteditor erstellen und mit Variablen versehen kann. So ist es ein Leichtes, eine Vielzahl ähnlicher Briefe mit geringem Aufwand zu erstellen. Dies weiß sicher jeder zu würdigen, der schon einmal eine große Zahl von Einladungen oder Werbebriefen mit persönlicher Ansprache erstellen mußte.

Die Bedingungen und die Bildschirmmasken lassen sich durch die Verwendung der entsprechenden Module individuell an die jeweiligen Erfordernisse anpassen. Man kann zum Beispiel das Aussehen der Karteikarten je nach Verwendungszweck beliebig umgestalten.

Wird durch die Eingabe der Daten ein gewisser Auslastungsgrad erreicht, kann es notwendig werden, daß man mit dem Reorganisationsmodul seine Daten neu ordnen muß. Dieses Modul komprimiert die Daten und erzielt damit eine optimale Speicherplatzausnutzung. Angesichts der geringen Diskettenkapazität des TI 99/4A ist eine solche Reorganisation außerordentlich praktisch.

Alle Module sind übrigens menügesteuert, wodurch sich der Benutzer recht leicht im Programm zurechtfindet. Als zusätzliche Unterstützung wechseln die Bildschirmfarben bei bestimmten Operationen oder fehlerhaften Eingaben. Hundertprozentig abbruchsicher bei Falscheingaben ist Damast allerdings nicht, so daß man Vorsicht walten lassen muß, um nicht eventuell Daten, die zwar eingetippt, aber noch nicht abgespeichert sind, zu verlieren.

Damit der Anwender die ganzen Möglichkeiten auch voll ausschöpfen kann, wird eine umfangreiche Bedienungsanleitung mitgeliefert. Diese dürfte aber vor allem Neulinge auf diesem Gebiet vor nicht geringe Probleme stellen. Einzelne Teile werden nämlich sehr kompliziert beschrieben. Auch mit Beispielen ging man sehr sparsam um. Gerade bei diesem wichtigen Punkt hätte man (auch im Hinblick auf den relativ hohen Preis) ruhig anwenderfreundlicher arbeiten können. Es wäre besser, wenn der Hersteller etwas mehr Rücksicht auf Anfänger in der Datenverarbeitung nehmen würde.

Ein weiterer kritischer Punkt ist die Verarbeitungsgeschwindigkeit. Da erstens das Programm in Extended Basic geschrieben ist und nur bei den Unterroutinen mit Maschinensprache arbeitet, und zweitens immer nur ein Modul auf einmal eingelesen wird (die anderen verbleiben auf Diskette), kommt es zwangsläufig immer wieder zu Wartezeiten während der Abarbeitung. Diese bleiben jedoch noch im Rahmen des vertretbaren; die Geschwindigkeit von Damast ist gerade noch ausreichend.

Zusammenfassend kann man Damast als gelungenes Datenverarbeitungsprogramm sehen, dessen einziger gravierender Nachteil lediglich in einem für meine Begriffe zu kompliziert geschriebenen Bedienungshandbuch liegt. Aber das kann sicher noch verbessert werden.

Für die Nutzung des Softwarepakets benötigt man den TI 99/4A, das Extended Basic, die 32-KByte-RAM-Erweiterung, 1 bis 3 Diskettenlaufwerke und 1 bis 3 Drucker (optional).

Die Preise für die einzelnen Programmteile: Kartenverwaltung und Analysemodul 218 Mark, Generierungs- und Reorganisationsmodule 89 Mark, Druckprogramm mit Drucklistengenerator und Selektionsgenerator 129 Mark und Texteditor Expert mit Verknüpfungsprogramm 129 Mark.

(Wolfgang Czerny)

# Feinkost für den TI99/4A

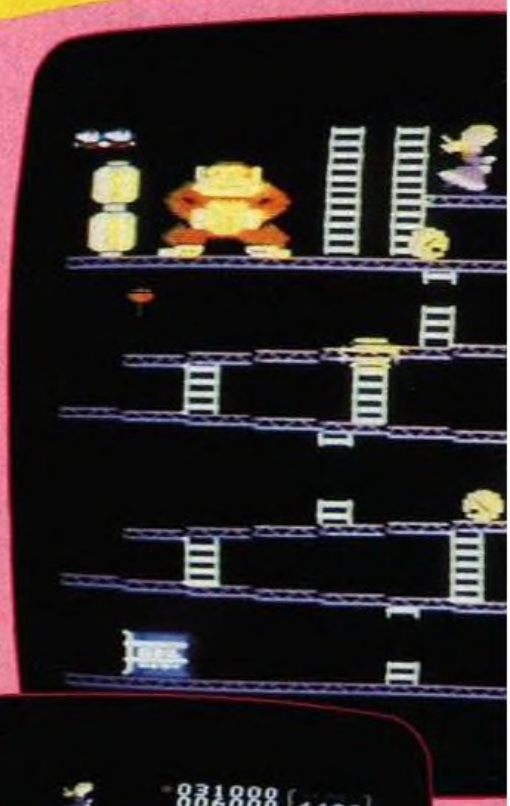


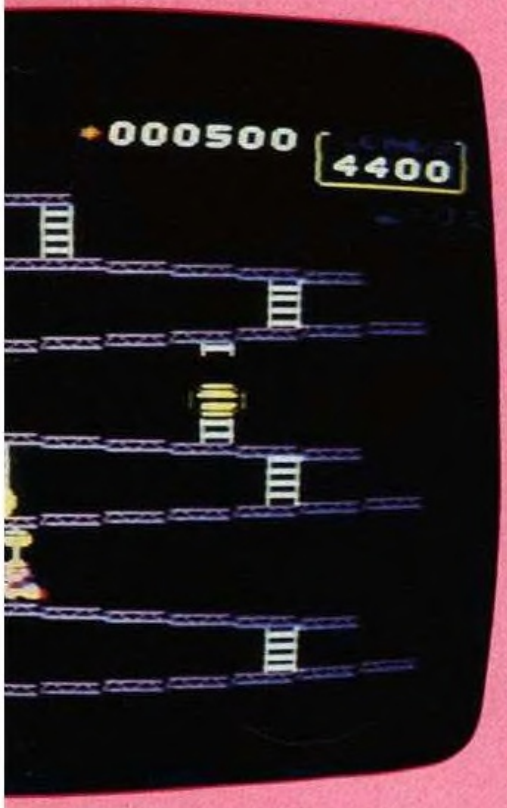
Bild 1. Pac Man — gelungene Umsetzung des Superspiels für den TI 99/4A

**Anzeigen versprechen in letzter Zeit TI 99/4A-Spieleversionen bekannter Spielhallenhits. Was taugen Sie?**

Natürlich interessiert es viele TI 99-Besitzer, wie gut es Atari gelungen ist, bekannte Spielhallenhits wie Pac Man, Defender, Donkey Kong und all die anderen für den 99er umzusetzen. Vor allem, wenn man an die Texas Instruments-Spielmodule denkt, die zum großen Teil nicht überzeugen konnten. Parsec war in diesem Fall eine der wenigen Ausnahmen. Also kann man wohl auch von den umgesetzten Programmen nicht viel erwarten —, könnte man denken. Doch welche Überraschung, wenn man ein Atari-Modul einschiebt und den Computer einschaltet. Anfangs das von TI gewohnte Titelbild, dann die Möglichkeiten, TI-Basic oder eben das Atarispiel zu wählen, beispielsweise Pac Man. Hat man sich nun für den gelben Vielfraß entschieden und Taste 2 gedrückt, so erscheint in leuchtend gelber Schrift das Pac Man-Titelbild, das noch durch seine Schlichtheit enttäuscht. Nachdem man aber geklärt hat, ob man allein oder lieber zu zweit spielen will und bei welchem Schwierigkeitsgrad (wählbar von 1 bis 9) man beginnen möchte, kann der Spaß endlich anfangen. Was nun auf dem Bildschirm erscheint, wird jedem TI-Benutzer zeigen, was eigentlich in dem kleinen, silbergrauen und unscheinbaren Kasten alles drinsteckt. Atarisoft zeigt hier Grafik in Arcadenqualität, die keinen Vergleich mit dem Original zu scheuen braucht.

Wird sich der erste gute Eindruck auch bestätigen? Es erklingt eine Melodie, die zwar nicht ganz der Pac Man-Einführungsmelodie entspricht, aber trotzdem nett gemacht ist. Und schon beginnt der kleine Allesfresser loszublubbern. Blubbern deshalb, weil es so klingt, als würde Mr. Pac Man eher ertrinken als Waffeln. Läuft unser Gelber hingegen über bereits leergefressene Wege, so ertönt die bekannte Sirene. Während Pac Man nun mehr oder weniger durch das Labyrinth irrt, kommen die grafisch perfekt nachgemachten Geister Inky, Pinky und Clyde ins Spiel, um die große Hetz-

jagd auf den gelben »Gummiball« zu starten. Hier bemerkt man sofort die Liebe der Atari-Programmierer zum Detail, denn genau wie beim großen Vorbild rollen die Augen der Gespenster bei jeder Richtungsänderung in die entsprechende Richtung. Auch sonst ist es vollkommen gelungen, dieses Superspiel für den TI 99/4A umzuschreiben. Hat man nämlich die erste Waffelladung vollständig abgeräumt, geht der ganze Spaß mit höherer Geschwindigkeit von vorne los. Natürlich erscheinen in jeder Runde die Bonusfrüchte, die erst nur 100, später aber bis zu 5000 Punkte einbringen. Im großen und ganzen kann man sagen, daß dieser Pac Man momentan zum Besten zählt, was für den TI 99/4A erhältlich ist. Grafik, Sound, Action — bei diesem Spiel stimmt einfach alles.



**Bilder 2 und 3.**
**Wer nicht um das Original trauert, wird an Donkey Kong für den TI 99/4A seine Freude haben.**

**Bild 4.**
**Defender — in der TI 99/4A-Version hat es nicht so viel zu bieten wie der große Bruder**

## Nicht ganz original, aber originell

Bei Donkey Kong sieht das schon ein wenig anders aus: Die Grafik entspricht nicht dem Original, was mit dem Videoprozessor TMS 9918A aber auch nicht möglich ist. Die Eisenträger sind bedingt durch den Bildschirm länger als bei der Automatenversion. Um dem Spiel mehr Arcadencharakter zu verleihen, hätte man ruhig auf die rote Eisenträgerfarbe zurückgreifen sollen, anstatt dem »Coleco-Vision-Lila« den Vorzug zu geben! Es sollte vielleicht erwähnt werden, daß diverse Spielhallentricks leider nicht anwendbar sind. Man hat zum Beispiel keine Möglichkeit, durch das bekannte »Handauflegen« zu verhindern, daß man während eines Leiteraufenthalts von herabkommenden Fässern erschlagen wird. Ferner gelingt auch der berühmtberüchtigte »Todessprung« nicht. Donkey-Kong-Freaks kennen garantiert auch den Hammer-Fließband-Trick, der in dieser Version leider auch nicht zum Punktesammeln verwendet werden kann.

Außerdem wird der Spielgenuß ein wenig durch die nervende Melodie getrübt, die aus dem Fernsehlautsprecher quillt. In diesem Punkt haben sich die Atari-Programmierer wenig um das Vorbild gekümmert, obwohl es sehr einfach gewesen wäre, die Einführungsmelodie zu verwenden, die jedem eingefleischten Donkey-Kong-Fan geläufig ist. Schade! Ist Bild 1 geschafft, so geht es mit Station 2 weiter, (also dem Bild mit den Eisenträgern) Übrigens: Keine Angst vor dem »Verfolgungsfeuer«, das unserem lieben Mario die Lust am Trödeln nehmen soll. Sobald ein blaues Faß gegen die Öltonne schlägt, erscheint zwar eine Flamme, aber die macht keine Anstalten, die Verfolgung aufzunehmen. Es fällt sofort auf, daß die amerikanische Bildfolge 1,4,1,3,4,1,2,3,4,1... bevorzugt wurde, was zwar vom Gerüstaufbau absolut unlogisch ist, insgesamt aber die Spannung erhöht, da man einige Zeit braucht, um das letzte Bild mit den Fließbändern zu erreichen. Bild 2, welches unserem viertem Bild entspricht, ist das Bild mit den gelben Eisennieten. Hut, Telefon und Regenschirm sind vorhanden und wie beim Video-Automaten angeordnet.

Man fühlt sich schon fast wie in der Spielhalle, doch dann eine kleine Enttäuschung: Das dritte Bild mit den Aufzügen entspricht schon nicht mehr ganz den Erwartungen, denn man hat sofort das ungute Gefühl, daß irgend etwas fehlt...Klar!!! die Springfedern, die unserem Zimmermann das Leben schwer machen. Bild 2, welches unserem viertem Bild entspricht, sind einfach weggelassen worden. Dafür ist die Flamme am oberen rechten Bildrand um einiges aggressiver als gewohnt. Die große Ernüchterung kommt aber erst mit Bild 4: Die Laufbänderstation sieht ziemlich mißglückt aus. Die Eisenpfannen sind so klein, daß man sie kaum erkennen kann, die Farben sind äußerst ungüstig gewählt (grüne Fließbänder, blaue Pfannen, rote Flammen...einfach furchtbar!), und außerdem fehlen die unterbrochenen Leitern, die es Mario nur zu bestimmten Zeitpunkten erlauben, das oberste Stockwerk zu erreichen. Es ist möglich, daß in diesem Fall zuwenig Speicherplatz vorhanden war, um ein grafisch vollendetes Bild zu gestalten.

Vielleicht wäre es besser gewesen, ein Bild wegzulassen und dafür ausgefeilte Gags und hochwertige Grafiken bei den anderen Stationen zu bringen. Wer allerdings nicht allzuviel Wert darauf legt, ein möglichst originalgetreues Spiel zu haben, der ist mit diesem Spiel sehr gut bedient. Vom Spielgeschehen her ist es nämlich ein gut gemachtes Videospiel, das sich in dieser Beziehung weitgehend an das Original hält.

# Feinkost für den TI 99/4A

Die Grafik ist nicht ganz wirklichkeitsgetreu, für Texas Instruments dennoch bestechend. Wer bisher nur die TI-Spielmodule gewohnt ist, wird von diesem Spiel wahrscheinlich begeistert sein, denn insgesamt kommt doch eine gewisse Spielhallenatmosphäre auf. Also rechtfertigt auch dieses Spiel genauso wie Pac Man den hohen Preis von 129 Mark.

## Ein anderes Beispiel: Defender

Hier muß gleich gesagt werden, daß der TI 99/4A nur einen 16-K-Speicher besitzt, der Original-Defender aber mit einem 26-K-Speicher ausgestattet ist. Klar, daß dieses Spiel auf dem TI eine Nummer kleiner ausfällt und nicht ganz soviel zu bieten hat, wie der große Defender-Bruder. Bei diesem hat man es nämlich mit sechs Bedienungselementen zu tun, dem Joystick, dem Reverseknopf, dem Hyperspace, dem Firebutton, dem »Thrust« und den »Smartbombs«. Beim TI hat man nur den Joystick, mit dem sich das Raumschiff in vier Richtungen steuern läßt, den Feuerknopf und die Tastatur. Durch Drücken der Space-Taste kann man seine Smartbombs abwerfen, hingegen verschwindet man bei allen anderen Tasten im Hyperspace.

Beschleunigen kann man seinen Star-Fighter, indem man den Joystick nach links oder rechts bewegt. Dabei verändert sich aber auch sofort die Flugrichtung des Raumschiffes und exakte Wendemanöver sind natürlich nicht mehr möglich; genausowenig wie automatisches MG-Feuer, was sehr schnell dazu führt, daß der Daumen zu schmerzen beginnt. Selbstverständlich gibt es auch beim kleinen Defender den Radarschirm am oberen Bildschirmrand, der einem aber nicht so hilfreich zur Seite steht wie beim Original, da die einzelnen Raumschiffe nur schwer zu erkennen sind, beziehungsweise man nur durch langes Üben herausfinden kann, wo die Lander gerade unsere Humanoiden entführen wollen. Dennoch ist der Scanner eine gute Hilfe.

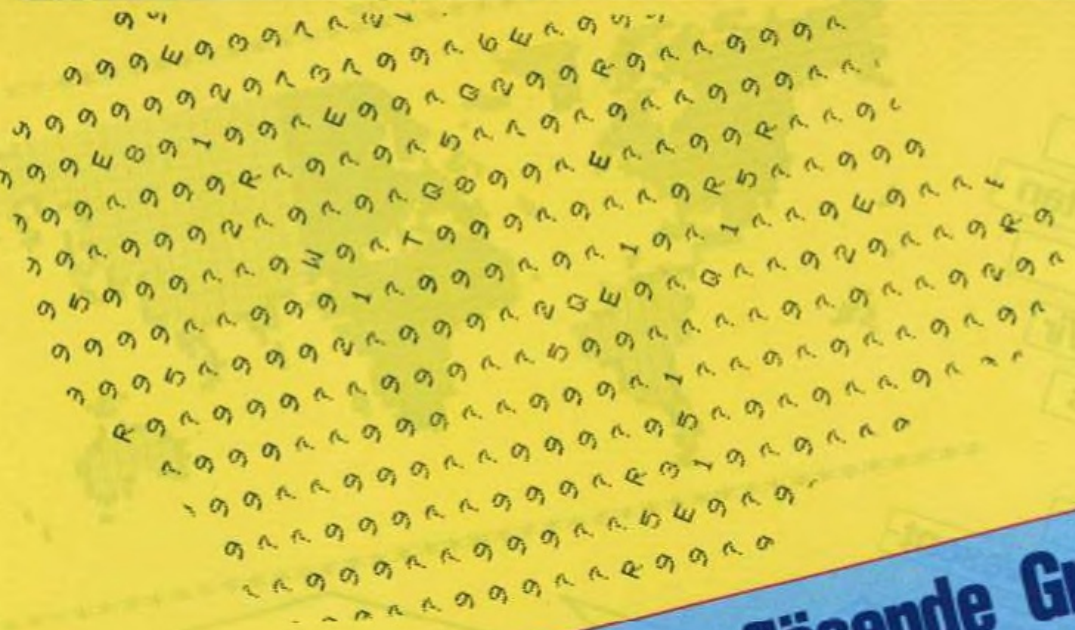
Ansonsten sind die bekannten Raumschiffe wie die Baiter, die Schwärmer, die Bomber, die Pods, die Mutanten und die weißen Mutterschiffe grafisch gut gestaltet. Wenn man sich also in Erinnerung ruft, daß der TI 99/4A einen verhältnismäßig kleinen Speicher hat, so wird man feststellen, daß Atari mit Defender ein Spiel geschaffen hat, das an Geschwindigkeit und Action keine Wünsche offen läßt. Selten gab es ein Spiel, das so komplex war wie dieses. Sicherlich ist es für den Anfänger sehr frustrierend, nach Spielende feststellen zu müssen, daß man nur so um die 5000 Punkte zustande gebracht hat; dafür hat man aber auch ein Spiel, das man auch noch nach vielen Spielstunden gerne in den Modulschacht einschiebt, um einen Weltraumkrieg zu starten. Parsec dagegen war ein Spiel, das man locker aus der Hüfte spielen konnte, wenn man erst einmal erkannt hatte, nach welchem Grundmuster man vorzugehen hatte. Es war dann kein Problem, den Highscore zu übertreffen und jede Menge Punkte zu machen. Im Gegensatz zu Defender, bei dem man sicher ein paar Trainingsstunden mehr braucht, um es einigermaßen zu beherrschen. Vorausgesetzt man nähert sich mal der 100000 Punktegrenze, sind einem Blasen an Daumen und Fingern sicher, da man den Joystick zwangsläufig so behandelt, als ginge es darum, ihn zu zerbrechen. Zusammenfassend kann man sagen, daß auch dieses Spiel die Anschaffung wert ist.

## Arcaden-Hits für den TI 99/4A bestechen durch ihre Grafik

Betrachtet man nun das gesamte Atari-Sortiment, fällt auf, daß sich die Atarisoft-Programmierer viel Mühe gegeben haben, um die Arcadenspiele erst für die Atari-Computer und dann auch noch für den TI 99/4A umzusetzen. Sämtliche Spiele bestechen durch ihre Grafikqualitäten und ihren doch recht hohen Spielreiz, der auch nach langer Zeit noch erhalten bleibt. Beim Sound hat man wohl weniger Sorgfalt walten lassen, was an sich auch nebensächlich ist, wenn man nur an die bisher angebotene Software für diesen Computer denkt. Laut Atarisollen noch weitere Spiele, zum Beispiel Dig-Dug, Centipede, Robotron 2084 und Stargate lieferbar sein; nicht einmal ein gutes Video-Spiel-Haus im Zentrum Münchens konnte naheres zum Erscheinungsdatum dieser interessanten Spiele sagen. Atari hingegen kündigte gleich weitere Arcadenhits für den TI 99/4A an. Für die Monate Februar bis April plant man unter anderm die Anlieferung von Joust, Jungle-Hunt, Pole Positione und Moon Patrol. Ferner plant der Hersteller noch eine Veröffentlichung von Lernspielen für die ganze Familie. Man darf also gespannt sein, was von Atari noch alles auf die TI-Besitzer zukommt. Es ist nämlich sehr erfreulich, daß Atari nicht wie andere Firmen (zum Beispiel Thorn Emi) einen Rückzieher gemacht hat, als bekannt wurde, daß Texas Instruments die Produktion des TI 99/4A eingestellt hat. Man kann also nur hoffen, daß Atari weitermacht und die angekündigten Spiele auch wirklich auf den Markt bringt.

(Volker Botschen)

Volker Botschen, 1965 in Gronau/Westfalen geboren, besucht derzeit die Adalbert-Stifter-Realschule in München. Sein Ziel: Groß- und Außenhandelskaufmann im Bereich der Unterhaltungselektronik. Aber sein Computerhobby (TI 99/4A, CBM 4032) ist nicht Volkers einzige Leidenschaft. Er treibt intensiv Sport (Volleyball) und fährt sehr gern Motorrad.



So sieht ein kleines Stück der Weltkarte auf Seite 57 in der Eingabe aus.

# Hochauflösende Grafik auf dem TI 99/4A

**Mit tollen Bildschirm-Darstellungen imponiert man Freunden und Bekannten. Die viele Arbeit, solche Grafiken in Extended Basis zu erzeugen, wird durch das folgende Programm wesentlich verringert.**

wird das Grafikzeichen auf dem Seikosha-Drucker ausgegeben. Nach diesem Verfahren kann man beliebige Zeichen für den Drucker konstruieren. Im folgenden Programm wurden einige der Grafikzeichen für den Drucker neu definiert. Diese Zeichen können besonders nützlich für Histogrammdarstellungen sein.

## Programmaufbau

Die erste Zeile beinhaltet in einem REM-Befehl die zu druckende Zeile und schließt mit einer Leerstelle ab. In den Zeilen 100 bis 190 wird ein Feld angelegt, in dem alle Zeichen vorübergehend abgespeichert werden.

Als Beispiel ist nach dem Rasterverfahren eine Weltkarte in 32 Zeilen und 66 Spalten kopiert worden. Allerdings ist die Zeichendefinition nicht abgedruckt, diese kann sich jeder selbst leicht erstellen. Zu dem Beispielprogramm bleibt noch folgendes zu sagen: In den Zeilen ab 1300 wird zeilenweise in einer Doppelschleife der Ausdruck vorbereitet, in denen die in A$ (32,66,1) gespeicherten Zeichen decodiert und die ab Zeile 900 definierten Grafikzeichen ausgedruckt werden.

Und jetzt viel Spaß bei der Erstellung dieser Super-Weltkarte!

(Dirk Weidmann)

```
10 ! HOCHAUFLOESENDE GRAPHIK
     AUF DEM TI-99/4A
20 ! von Gerald Spiegel
          Schubertstr.5
           7640 Kehl am Rhein
30 SUB FEIN(V,W)
40 W=INT(W):: V=INT(V):: H$="1
   23456789ABCDEF" :: VG=INT(V
   /8):: WG=INT(W/8):: VR=V-VG
   *8+1 :: WR=W-WG*8+1 :: CALL
    GCHAR(VG+1,WG+1,Q):: Z=VR*
   2+INT(WR/4.1)-1
50 W1=INT((WR/4.1-INT(WR/4.1))
   *4.1+.5):: IF Q=32 THEN K=K
   +1 :: Q=K+32 :: C$=RPT$("0"
   ,16)ELSE CALL CHARPAT(Q,C$)
60 C$=SEG$(C$,1,Z-1)&SEG$(H$,P
   OS(H$,SEG$(C$,Z,1),1)OR 2^(
   4-W1),1)&SEG$(C$,Z+1,15)::
    CALL CHAR(Q,C$):: CALL HCHA
   R(VG+1,WG+1,Q):: SUBEND
```

Listing zum Unterprogramm »Fein«

Mit diesem Unterprogramm ist es möglich, Punkte des Rasters 192 x256 über die Anweisung CALL FEIN (X,Y) direkt anzusteuern. Das Programm gliedert sich in drei Teile. In der Zeile 40 wird die genaue Position des Grafikpunktes im Grobraster (24 x 32 Punkte) und im zugehörigen 8 x 8-Feld festgestellt. Zeile 50 errechnet das zu definierende ASCII-Zeichen. Schließlich wird in Zeile 60 der Hex-Code zusammengestellt, das Zeichen definiert und in das Grobraster eingesetzt.

## Wichtig: Unterprogramm verschieben

Es empfiehlt sich, nach dem Eintippen des Unterprogramms dieses mit »RES 1000« in einen höheren Zeilennummernbereich zu verschieben und so Platz für das Hauptprogramm zu schaffen, da die Verwendung von höheren Zeilennummern für Zeilen des Hauptprogramms als für jene des Unterprogramms nicht zulässig ist.

Die Anweisung CALL FEIN (X,Y) kann beliebig oft im Hauptprogramm verwendet werden, wobei X die Reihen und Y die Spalten des Rasters darstellen. Der Ursprung befindet sich in der linken, oberen Ecke. Es ist darauf zu achten, daß die Koordinaten im Bereich des Rasters bleiben, weil ansonsten das Programm unterbrochen wird. Wenn die Feinauflösung zu großflächig verwendet wird, reicht der Zeichenvorrat des Computers eventuell nicht aus (110 Zeichen), und das Programm wird ebenfalls mit Fehlermeldung gestoppt.

## Hilfe für Funktionsdarstellungen

Bei der Darstellung der Funktionen ist die hochauflösende Grafik eine wertvolle Hilfe; sie kann aber auch anderweitig vielfach verwendet werden.

(Gerald Spiegel)

# Joystick-Abfrage in Maschinensprache

**Das Listing »VC 20-Routine zur Joystick-Abfrage« in Happy-Computer, Ausgabe 1/84 hat mich dazu bewogen, auch meine Version einer Joystick-Abfrage in Maschinensprache einmal vorzustellen.**

VC 20

```
.,  5E9A LDA $9111
.,  5E9D PHA
.,  5E9E LSR
.,  5E9F LSR
.,  5EA0 AND #$07
.,  5EA2 STA $FB
.,  5EA4 PLA
.,  5EA5 AND #$20
.,  5EA7 ASL
.,  5EA8 ASL
.,  5EA9 ORA $FB
.,  5EAB STA $FB
.,  5EAD LDX #$7F
.,  5EAF STX $9122
.,  5EB2 LDA $9120
.,  5EB5 LDX #$FF
.,  5EB7 STX $9122
.,  5EBA AND #$80
.,  5EBC LSR
.,  5EBD LSR
.,  5EBE LSR
.,  5EBF LSR
.,  5EC0 ORA $FB
.,  5EC2 ORA #$70
.,  5EC4 EOR #$FF
.,  5EC6 TAY
.,  5EC7 LDA #$00
.,  5EC9 JMP $D391

.ASSEMBLER-LISTING
```

Assembler-Listing »Joystick-Abfrage«

Das Programm benötigt 50 Bytes und ist im Speicher voll verschiebbar. Außerdem weist diese Routine gegenüber der in der Januar-Ausgabe einige entscheidende Vorteile auf.

1. Der Aufruf erfolgt über die Basic-Funktion USR(X) und der »Wert« des Joysticks kann direkt einer Variablen zugewiesen werden (»A = USR(0)«).

2. Die Werte, die vom Maschinenprogramm geliefert werden, lassen sich wesentlich leichter merken. Die Werte sind: Nord = 1, Süd = 2, West = 4, Ost = 8, Feuer = 128. Die Werte der Zwischenrichtungen sind die Summe der Einzelrichtungen. Eine Bewegung nach Nord und Ost gleichzeitig ergibt den Wert 1 + 8 = 9. Auch der Feuerknopf wird auf diese Weise addiert. Aufgrund dieser Ergebnisse lassen sich die Werte besser auswerten. Zum Beispiel ist hier auch eine bitweise Weiterverarbeitung leicht möglich.

3. Diese Routine läßt sich direkt in Programme integrieren, welche für die Joystickabfrage des Grafikmoduls von Commodore geschrieben sind. Lediglich der Aufruf (A = RJoy(0)) ist

# Tastaturabfrage

**Mancher Programmtitel ist mit einer netten Melodie verziert, die auf das nachfolgende Programm einstimmen soll. Besonders bei Spielen sind solche Musiktitel beliebt. Zum Starten des Programms muß nur noch eine beliebige Taste gedrückt werden.**

Der TI 99/4A besitzt gute musikalische Möglichkeiten. Man sollte sie auch in eigenen Programmen für nette kleine Vorspann-Melodien nutzen. Bei vielen selbstgestrickten Programmen fällt aber auf, daß diese musikalischen Möglichkeiten nicht optimal mit der Bedienerführung verbunden sind. Wohl wird die Melodie zu Programmbeginn abgespielt, aber die Aufforderung, eine beliebige Taste zu drücken, damit das eigentliche Programm beginnen kann, hält den Anwender nicht selten zum Narren. Drückt er nämlich — ganz nach Anweisung — eine Taste, rührt sich überhaupt nichts. Erst wenn er mit großer Ausdauer längere Zeit die Taste gedrückt hält, reagiert der Computer.

Die Ursache ist eine schlechte Implementierung des Befehls für die Tastatur-Abfrage (CALL KEY). Wir zeigen hier eine einfache Routine, die eine schnelle Reaktion des Programms während der ganzen Dauer der Melodie erlaubt. Dabei hilft uns der Umstand, daß beim TI 99/4A der Tongenerator unabhängig vom Programmablauf arbeitet,

durch »A = USR(0)« zu ersetzen. Das Basicladeprogramm in der abgedruckten Form speichert das Unterprogramm in das obere Ende des RAM-Bereichs des VC 20. Hierzu wird in Zeile 120 die Adresse der letzten RAM-Zelle ermittelt und hiervon 53 (die Anzahl der Bytes, die das Maschinenprogramm benötigt) abgezogen. In Zeile 130 wird das RAM-Ende mit diesem errechneten Wert neu gesetzt. Der Befehl »CLR« in Zeile 140 bewirkt nun ein Korrigieren der Zeiger für die Verwaltung der Arrays.

Die Anfangsadresse des Programms muß nun neu ermittelt werden. Dies erledigt Zeile 150. Die Einsprungadresse ist also RAM-Ende + 2. Nun wird das Maschinenprogramm in den Speicher »gePOKEt« und zu guter Letzt die USR-Sprungadresse mit der Einsprungadresse des Maschinenprogramms gesetzt. Hiermit ist das Ladeprogramm beendet und der Joystick kann nun in der oben beschriebenen Weise abgefragt werden. Auch kann bei dieser Joystickabfrage eine Taste des Recorders betätigt werden. Das Programm filtert das entsprechende Bit ohnehin aus.

(Jürgen Reinert)

```
10 rem"  ************************
20 rem"  * Joystickabfrage in *
30 rem"  *   Maschinensprache  *
40 rem"  * mit direkter Wert- *
50 rem"  * zuweisung          *
60 rem"  * (C) 1983           *
70 rem"  *   by  J.Reinert    *
80 rem"  ************************
90 :
100 :
110 :
120 es=peek(55)+256*peek(56)-53
130 poke55,es-(int(es/256)*256)
140 poke56,es/256:clr
150 es=peek(55)+256*peek(56)+2
160 forb=0to49:reada:pokeb+es,a:s=s+a:next
170 ifs<>5461thenprint"■2data error■9■":end
180 poke0,76
190 poke1,es-(int(es/256)*256)
200 poke2,es/256
210 end
220 data173,17,145,72,74,74,41,7
230 data133,251,104,41,32,10,10,5
240 data251,133,251,162,127,142,34
250 data145,173,32,145,162,255,142
260 data34,145,41,128,74,74,74,74
270 data5,251,9,112,73,255,168,169
280 data0,76,145,211
290 :
300 :
310 :
320 rem" ■ = crsr down
330 rem" ■ = rvs on
ready.
```

**Listing des Programms »Joystick-Abfrage«**

# ...nitten im Lied

TI 99/4A

```
50 CALL LIED
60 PRINT "OK" :: END
100 REM LIED-ROUTINE MIT TASTATUR-ABFRAGE
110 SUB LIED
120 !
130 RESTORE 170 :: FOR LOOP=1 TO 16
140 READ TON,DAUER :: CALL SOUND(DAUER*30,TON,0)
150 CALL KEY(3,K,R):: IF K)-1 THEN SUBEXIT
160 NEXT LOOP :: GOTO 130
170 DATA 175,14,262,15,233,5,220,5,196,5,349,14,262,15,233,5
180 DATA 220,5,196,5,349,14,262,15,233,5,220,5,233,5,196,15
190 SUBEND
```

**Listing für Tastaturabfrage während einer Melodie**

solange ein einmal begonnener Ton andauert. Im Listing ist ein kleines Beispiel mit der Melodie aus »Krieg der Sterne« abgebildet. (Die Melodie ist natürlich rein willkürlich gewählt.)

Der eigentliche Trick liegt darin, daß ein Sound-Befehl aufgerufen wird und gleichzeitig (noch während die Tonausgabe läuft) eine Tastaturabfrage mit CALL-KEY stattfindet. Das eigentliche Programm hierfür ist als Unterprogramm (in Extended Basic) in den Zeilen 110 bis 190 formuliert worden. Die Tondaten in Zeile 170 und 180 können je nach Geschmack an andere Melodien angepaßt werden. Diese Routine ist auch in normalem TI-Basic möglich, wenn an Stelle der Doppelpunkte jeweils eine neue Zeile geschrieben wird. Zeile 190 kann dann übrigens entfallen. Die Zeile 150 aber wird zu »...THEN RETURN«, Zeile 50 zu »GOSUB 130« und Zeile 110 zu »END«. Wichtig an diesem Programm ist nur, daß die Dauer der Töne nicht kürzer gewählt wird, als die Tastatur-Abfrage dauert, da sonst Lücken in der Melodie auftreten. 100 Millisekunden für die Tonlänge reichen normalerweise aus.

Selbstverständlich kann die kleine Routine auch an den Programmanfang gestellt werden (ohne Sub-Programm-Definition). Dann muß allerdings in Zeile 150 als Sprungadresse die Programmzeile stehen, in der das Hauptprogramm beginnen soll. (lg)

NEW LINE getastet werden.

Vorschlag B wird dann benötigt, wenn die eingegebenen Daten aus mehreren Zeichen bestehen und weiterverarbeitet werden sollen. Hier wird dann das erste Zeichen der Zeichenkette zur Entscheidungsfindung herangezogen. Die Länge der Zeichenkette spielt keine Rolle, Zeile 12 verhindert ENTER beziehungsweise NEW LINE ohne vorherige Zeicheneingabe. Diese Zeile darf auf keinen Fall entfallen.

Mit dem Vorschlag B können natürlich nicht nur einzelne Zeichen, sondern beliebig lange Zeichenketten abgefragt werden. So könnten die Zeilen 13, 14, 15 oder 16 auch lauten:

IF B$ = "YES" THEN ....
oder
IF B$ = "NEIN" THEN ....

Dies kann zum Beispiel zur Eingabe von Schlüsselworten verwendet werden, die die Nutzung des Programms nur dem gestatten, der das richtige Kennwort fehlerfrei eingibt.

Lösungsmöglichkeit C wird nötig, wenn eine numerische Eingabe erwartet wird (nur Zahlen).

INPUT C führt zum Programmabbruch, wenn zum Beispiel ein Buchstabe getastet wird. Eine String-Variable hingegen akzeptiert jede Eingabe. In der FOR-NEXT-Schleife wird nun geprüft, ob nur Werte zwischen 0 und 9 eingegeben wurden. Wenn nicht, geht es zurück zur Dateneingabe.

Starten Sie mein Demonstrationsprogramm mit RUN 1, RUN 11, RUN 101 beziehungsweise RUN 1001 und probieren Sie die jeweiligen Wirkungen aus.

Die Programmzeilen 4, 6, 14, 16, 103 und 105 entfallen beim ZX81. Bei dem Spectrum ist in die Anführungszeichen der jeweilige Kleinbuchstabe zu setzen.

Natürlich ist es in einem richtigen Programm besser, die Eingaberoutinen in echte Unterprogramme zu verwandeln. Dazu sind die Zeilen 9, 19, 109 und 1009 in »RETURN« umzuändern und die Routinen vom Hauptprogramm mit GOSUB aufzurufen. Dieses Verfahren spart sogar wesentlich Speicherplatz, wenn Sie ein eingabeintensives Programm haben. Statt INPUT wird GOSUB X programmiert.

Die aufgeführten Möglichkeiten sind sicherlich nicht vollständig, jedoch bestimmt hilfreich und nützlich.

Ich denke, daß auch Benutzer anderer Computer die Lösungsvorschläge, wenn auch in abzuwandelnder Form, nutzen können und sollten.

(Manfred Kotting)

**Viele TI 99/4A-Programme benützen die Joysticks als Eingabegeräte. Bei Spielen erleichtern diese Steuerknüppel den Spielablauf. Aber nicht jeder TI-Benutzer besitzt welche. Die kleine Routine »Joystick« stellt einen allgemein gültigen Ersatz für die Joysticks dar.**

# Auch ohne Stick viel »Joy«

W R E B D

X Z C

**Tabelle der Tasten mit Joystick-Funktionen:**

Das Unterprogramm »JOYSTICK« (in Extended Basic geschrieben, aber leicht auf TI-Basic umsetzbar) kann an Stelle der JOYST-Aufrufe in ein Programm eingefügt werden. Die Parameter sind wie beim Originalbefehl anzugeben. Die zurückgelieferten Werte entsprechen denen eines Joysticks. Nur die Reaktion ist etwas langsamer. Als Ersatz für die Steuerknüppel der Fernsteuerung dienen die Tasten E, D, X, S, W, R, C, Y (für Joystick 1) und O, L, <, K, I, P, > und M (für Joystick 2). Die Zeilen 100 bis 150 sind nur zur Demonstration des Umsetzungsprogramms nötig.

Wichtig ist, daß durch die Tastaturabfrage die Tastatur des TI 99/4A in einen Modus umgeschaltet wird, die ausschließlich Großbuchstaben erlaubt. Wenn Sie also anschließend wieder Kleinbuchstaben erzeugen wollen, müssen Sie Zeile 170 so formulieren: »170 CALL KEY(3,K,R)::Call KEY(0, R, R):: IF K=—1 THEN SUBEXIT«. Die Routine selbst muß selbstverständlich an das Ende des betreffenden Hauptprogramms gestellt werden. Wer ein Diskettenlaufwerk hat, kann sich die Subroutine mit Merge auf Diskette abspeichern und bei Bedarf hinzumischen. Im Hauptprogramm müssen allerdings erst alle JOYST-Aufrufe in JOYSTICK-Aufrufe verwandelt werden. Die Variablen können bleiben wie sie sind.

(lg)

```
100 CALL CLEAR
110 X,Y=0 :: CALL JOYST(1,X,Y):: DISPLAY AT(1,1):"ORIGINALWERT: ";X;Y
120 X,Y=0 :: CALL JOYSTICK(1,X,Y):: DISPLAY AT(2,1):"ERSATZWERT 1: ";X;Y
130 X,Y=0 :: CALL JOYSTICK(2,X,Y):: DISPLAY AT(3,1):"ERSATZWERT 2: ";X;Y :: GOTO
 110
140 !
150 !
160 SUB JOYSTICK(NR,X,Y)
170 CALL KEY(3,K,R):: IF K=-1 THEN SUBEXIT
180 IF NR=2 THEN 280
190 IF K=69 THEN Y=4 :: X=0 :: SUBEXIT
200 IF K=88 THEN Y=-4 :: X=0 :: SUBEXIT
210 IF K=83 THEN X=-4 :: Y=0 :: SUBEXIT
220 IF K=68 THEN X=4 :: Y=0 :: SUBEXIT
230 IF K=87 THEN X=-4 :: Y=4 :: SUBEXIT
240 IF K=82 THEN X,Y=4 :: SUBEXIT
250 IF K=67 THEN X=4 :: Y=-4 :: SUBEXIT
260 IF K=90 THEN X,Y=-4 :: SUBEXIT
270 X,Y=0 :: SUBEXIT
280 IF K=79 THEN Y=4 :: X=0 :: SUBEXIT
290 IF K=44 THEN Y=-4 :: X=0 :: SUBEXIT
300 IF K=75 THEN X=-4 :: Y=0 :: SUBEXIT
310 IF K=76 THEN X=4 :: Y=0 :: SUBEXIT
320 IF K=73 THEN X=-4 :: Y=4 :: SUBEXIT
330 IF K=80 THEN X,Y=4 :: SUBEXIT
340 IF K=46 THEN X=4 :: Y=-4 :: SUBEXIT
350 IF K=77 THEN X,Y=-4 :: SUBEXIT
360 X,Y=0 :: SUBEND
```

**Listing für die Ersatzroutine »JOYSTICK«**

## »Zauberschloß«

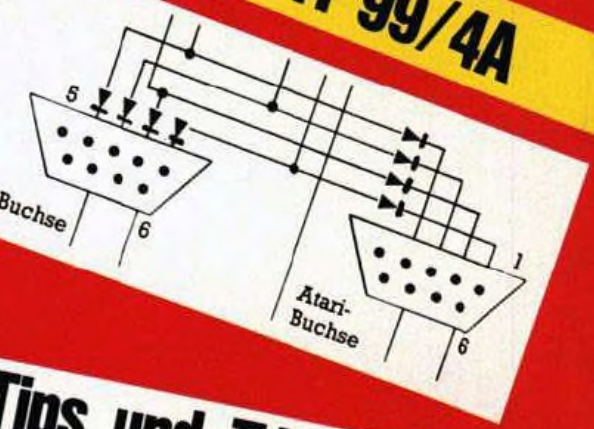
(Ausgabe 2/84, Seite 112)

Der hoffentlich letzte Fehler im Listing »Zauberschloß« ist uns von einem Leser gemeldet worden. In der Zeile 56112 muß die Variable BU in BU$ abgeändert werden, da sonst ein alter Spielstand von Diskette nicht korrekt eingelesen wird.

## Atari-Joystick für den TI 99/4A

(Ausgabe 6/84, Seite 115)

Beim Schaltplan für den Joystick-Adapter wurden versehentlich die Querstriche der acht Dioden einzuzeichnen vergessen. Hier also die verbesserte Schaltung:




## Hochauflösende Grafik für den ZX81, leider nicht ganz fehlerfrei

Der Fehlerteufel in der »Happycomputer« Ausgabe 6 hat leider auch die hochauflösende Grafik (Seite 50) für den ZX81 nicht verschont. Der komfortable HEX-Monitor (Bild 2, Seite 51) muß — um richtig komfortabel zu sein — in drei Zeilen geändert werden:

```
1020 IF Q$(1)<"0" OR Q$
(1)>"F" OR Q$(2)<"0"OR
Q$(2)>"F" THEN GO TO
1000
1100 IF F$="1" THEN GOTO
1000
1330 IF Q$(1)<"0" OR
Q$(1)>"F" THEN RETURN
```

In dem Maschinenprogramm zum Erzeugen der 1021 Byte langen REM-Zeile (Bild 3, Seite 51) ist noch eine »4« von ihm gestrichen worden. Die Zeile 7014 muß richtig lauten:

```
7014 CDB014 CALL 14BOH
```

Daß im Maschinenprogramm (auf Seite 53) die Zeile 438C nicht an ihrer richtigen Position, sondern neun Zeilen zu hoch stand, das haben Sie sicher bemerkt. In die Speicheradresse 16513 muß man den Wert 249 (entspricht dem Befehl »RAND«) poken. Die Befehle der Tabelle 1 auf Seite 50 funktionieren übrigens auch im FAST-Modus.

Durch Ändern der beiden Speicheradressen 439D (17309) und 439E (17310) kann eine wesentlich verbesserte Bilddarstellung erreicht werden:

| Adresse | alt | neu |
|---|---|---|
| 439D | A8 (168) | 94 (148) |
| 439E | AB (171) | 94 (148) |

(Frank P. Walter)

## Tips und Tricks für den Oric-1

(Ausgabe 5/84, Seite 112)

Beim Druck schlichen sich leider folgende Fehler ein: Seite 113, rechte Spalte, unterhalb der Mitte (»PRINT HEX $(PEEK(#A6))«) muß richtig lauten: »PRINT HEX$(DEEK (#A6))«.

Der Satz auf Seite 114, linke Spalte, oberhalb der Mitte (»...HIMEM auf #97FF statt auf #7FF«) muß richtig lauten: »...HIMEM auf #9EFF statt auf #97FF«.

(Heinz Dibold)

## Poker gegen den VC 20

(Ausgabe 3/84, Seite 122)

Im Programm ist die Zeile 3540 von Hand ergänzt worden. Dabei wurde übersehen, daß dadurch die Zeile mehr als 88 Zeichen aufweist. Das Problem läßt sich lösen, indem man hinter THEN zuerst eine Hilfsvariable setzt:

```
3540 IF(0)=A$(1)ANDA$(1)=
A$(2)ANDA$(2)=A$(3)AND
A$(3)=A$(4)THENX$=A$(4)
```

Nun folgt eine zusätzliche Zeile:

```
3545 IFX$=A$(4)THEN
SW=63+KT(0):SW$="STR
AIGT FLUSH":RETURN
```

Dasselbe gilt auch für die Zeilen 3660 und 3665:

```
3660 IF(0)=A$(1)ANDA$(1)=
A$(2)ANDA$(2)=A$(3)AND
A$(3)=A$(4)THENY$=A$(4)
3665 IFY$=A$(4)THENSW=
39+KT(0):SW$="FLUSH":
RETURN
```

(Josef Schneider)

## Sternen-Abenteuer

(Ausgabe 4/84, Seite 59)

Im Listing zu »Sternen-Abenteuer «, haben sich leider so viele Fehler eingeschlichen, daß eine vollständige Korrektur an dieser Stelle nicht möglich ist. Leser, die sich aber für ein korrektes Listing interessieren, können selbstverständlich kostenlos ein solches Listing bei uns anfordern; Karte genügt.

## Währungs-umrechnung

(Ausgabe 3/84, Seite 97)

Die Zeilen 12 und 14 müssen lauten:

```
12 INPUT "WAEHRUNG A
?";A$
14 INPUT "WAEHRUNG B
?";B$
```

(Dirk Bittner)

# Spiele

Ein Taucher, der mit den Cursortasten gesteuert wird, muß verschiedene Schätze bergen. Hat er einen gehoben, ertönt ein Signal und 100 Punkte werden Ihnen gut geschrieben. Aber Vorsicht: ein Hai, dem man geschickt ausweichen muß, will Ihren Taucher fressen, der sogar über den Bildschirmrand hinausschwimmen kann, was Ihnen einen Vorteil bietet, denn dem Hai gelingt dies nur sehr selten. So kann sich Ihr Taucher immer im Dunklen verstecken. Sind alle Schätze gehoben, wird eine kleine Siegesfeier veranstaltet. Mit jeder Runde wird das Spiel schneller und der Hai zielgenauer. Dem Taucher stehen übrigens fünf Leben zur Verfügung, um sein Ziel zu erreichen.

(Rolf Poos)

| | |
|---|---|
| Zeile 100-200 | Kopfzeilen |
| Zeile 230-470 | Firmenzeichen (kann weggelassen werden) |
| Zeile 490-530 | Notenwerte und Einlesen der Siegesfanfare |
| Zeile 560-800 | Definition der Grafik |
| Zeile 830-940 | Gestaltung des Spielfeldes |
| Zeile 990-1100 | Definition und Verteilung der Sprites |
| Zeile 1110-1270 | Spielerschleife<br>Steuerung des Tauchers durch Pfeiltasten<br>Abfrage, ob zwei Sprites sich berühren |
| Zeile 1290-1350 | Hai oder Schatz berührt? |
| Zeile 1360-1400 | Schatz geborgen |
| Zeile 1420-1540 | Taucher gefressen |
| Zeile 1550-1680 | Alle Schätze gehoben, Siegerrunde |
| Zeile 1690-1800 | Verloren, Hai freut sich |
| Zeile 1810-1880 | Neues Spiel? |
| Zeile 1890-2000 | Titel |
| Zeile 2010-2100 | Spielregeln |
| Zeile 2110-2150 | Spielende |

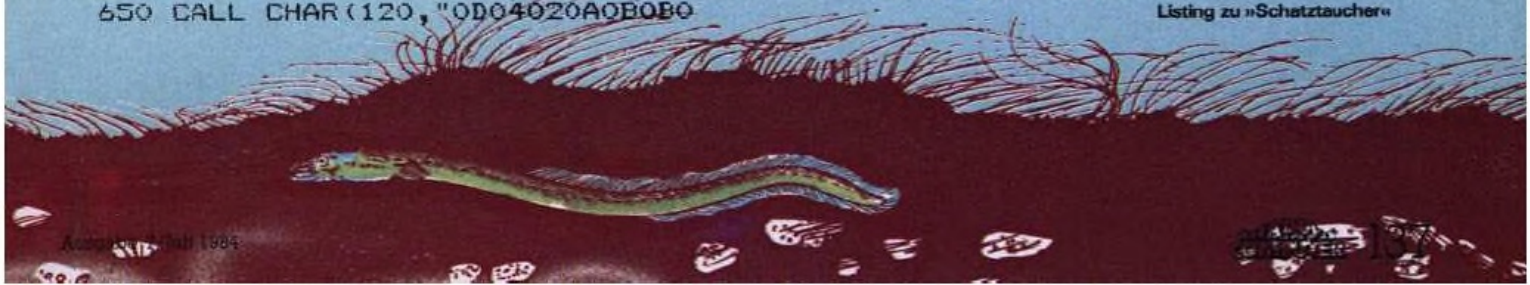
Tabelle zu »Schatztaucher«

```
100 REM**********************
110 REM*                     *
120 REM*   SCHATZTAUCHER     *
130 REM*                     *
140 REM* TI99/4A + EXT.BAS.*
150 REM*                     *
160 REM*  (C)1983 R.POOS     *
170 REM*                     *
180 REM**********************
190 REM
200 REM
210 REM
220 REM
230 CALL CLEAR :: CALL SCREEN(14)
240 CALL CHAR(96,"030C1020418181
    818242030604120C04E010080808
    0810204080000000000040A")
250 CALL CHAR(100,"0605080808101
    0100000000000000000B0A8A462
    00000000000")
260 CALL CHAR(104,"0000000000192
    9294A4A4A8A9295A6C400000000C
    0202040404080808304081 1")
270 CALL CHAR(108,"8C74042A1C08
    0808101010000000000011010311
    0103010101010000000")
280 CALL CHAR(112,"0000000000000
    000000000F8040404040000000000
    00000000000")
290 CALL CHAR(116,"0830C00006192
    9C906000000000000000000000001
    CF7141408000007 00")
300 CALL CHAR(120,"0000000000000
    0000000000000000000000000000
    000000000000000000")
310 CALL CHAR(124,"0000004080008
    0402010E0000000000000000000000
    0000000000000000000")
320 CALL MAGNIFY(4)
330 CALL SPRITE(#1,96,16,30,65)
340 CALL SPRITE(#2,100,16,60,65)
350 CALL SPRITE(#3,104,16,30,99)
360 CALL SPRITE(#4,108,16,60,99)
370 CALL SPRITE(#5,112,16,30,133
    )
380 CALL SPRITE(#6,116,16,60,133
    )
390 CALL SPRITE(#7,120,16,30,168
    )
400 CALL SPRITE(#8,124,16,60,168
    )
410 FOR I=1 TO 500 :: NEXT I
420 DISPLAY AT(16,9):"R O L F  P
     O O S"
430 DISPLAY AT(18,9):"S O F T W
    A R E"
```

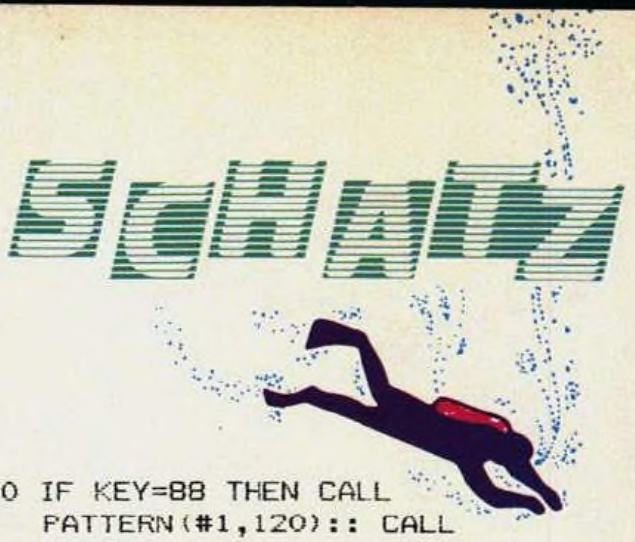
**Bergen Sie die Schätze der Titanic aus den Tiefen Ihres TI 99/4A. Als Taucher müssen Sie sich aber vor dem gefräßigen Hai hüten, der die Schätze bewacht.**

```
440 DISPLAY AT(20,9):"PRESENTS:"
450 FOR I=295 TO 193 STEP -50 ::
     FOR J=0 TO 30 STEP 5 :: CAL
    L SOUND(200+J,I,J,I*3,J,-3,J
    ):: NEXT J :: NEXT I
460 CALL DELSPRITE(ALL)
470 CALL CHARSET :: CALL CLEAR
480 REM SCHATZTAUCHER
490 DIM T1(20),T2(20),T3(20)
500 DATA 400,293,293,400,391,391
    ,400,493,493,200,391,391,100
    ,293,587,100,293,587,100,391
    ,493
510 DATA 100,293,440,100,391,493
    ,100,440,523,200,493,587,200
    ,523,659,200,493,587,100,440
    ,523
520 DATA 100,391,493,200,440,523
    ,200,369,587,400,391,493,100
    ,14000,14000
530 FOR I=1 TO 19 :: READ T1(I),
    T2(I),T3(I):: NEXT I
540 TAUCHER=5 :: V=6
550 RANDOMIZE
560 REM TAUCHER RECHTS
570 CALL CHAR(96,"000000000007C7
    F78F3FC09E010000000000000000
    C0D8E6FEFA4460F00F")
580 REM FISCH LINKS
590 CALL CHAR(100,"000000000F173
    F7FFF0F1F000000000000002060E
    1F1FBFEFEFBC1")
600 REM FISCH RECHTS
610 CALL CHAR(104,"00000406878FD
    F7F7FDF83000000000000000000F
    0E8FCFEFFF0F8")
620 REM TAUCHER OBEN
630 CALL CHAR(112,"0001010202010
    707070707000101030304C4A4C4C
    8D8F8C8C8D0D0D0504020B0")
640 REM TAUCHER UNTEN
650 CALL CHAR(120,"0D04020A0B0B0
    B13131F1B1323252320C0C080800
    0E0E0E0E0E08040408080")
660 REM TAUCHER LINKS
670 CALL CHAR(128,"0000000000031
    B7F5F22060FF000000000000000E
    0E3EFF1FC037980")
680 REM KRUG
690 CALL CHAR(132,"003C3E3D3D3E3
    C7E")
700 REM LAMPE
710 CALL CHAR(136,"18183C183C7E7
    E3C")
720 REM ZEPTER
730 CALL CHAR(140,"000018C37E7E3
    C18")
740 REM RING
750 CALL CHAR(108,"183C7EFFFF7E3
    C18000000000000000000000000
    00000000000000000000000")
760 REM WASSERPFLANZEN
770 CALL CHAR(116,"183C7E7E7E7E3
    C18")
780 REM BLUT
790 CALL CHAR(124,"0303010103000
    00000000000000000001CFEFFFFF
    FDF0301010000000")
800 CALL COLOR(11,6,1,12,3,1)
810 FOR I=1 TO 8 :: CALL COLOR(I
    ,16,1):: NEXT I
820 GOSUB 1830
830 REM SPIELFELD
840 CALL CLEAR
850 DISPLAY AT(1,2):"TAUCHER:";T
    AUCHER;"   PUNKTE:";PUNKTE
860 CALL VCHAR(19,7,116,5):: CAL
    L VCHAR(16,6,116,3):: CALL V
    CHAR(17,8,116,2)
870 CALL VCHAR(10,6,116,3):: CAL
    L VCHAR(13,5,116,3):: CALL V
    CHAR(13,11,116,2):: CALL HCH
    AR(12,1,116,2)
```

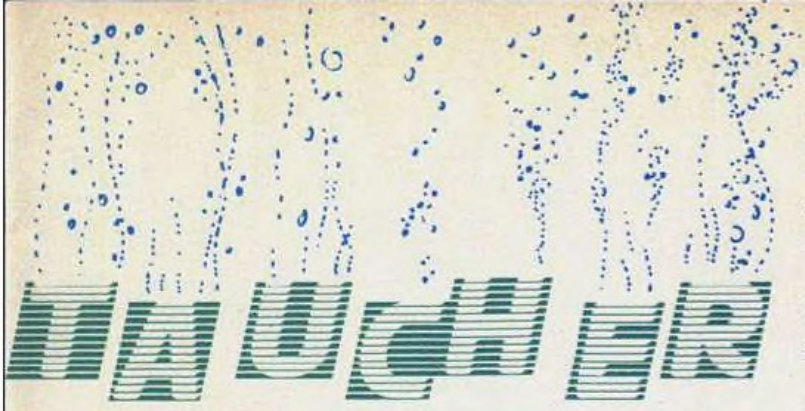
Listing zu »Schatztaucher«



```
880 CALL HCHAR(15,3,116,2):: CAL
    L HCHAR(16,9,116):: CALL HCH
    AR(15,10,116)
890 CALL VCHAR(13,21,116,11):: C
    ALL VCHAR(18,4,116,7):: CALL
     VCHAR(3,24,116,6)
900 CALL VCHAR(8,23,116,3):: CAL
    L VCHAR(11,22,116,2):: CALL
    VCHAR(10,20,116,3):: CALL VC
    HAR(9,19,116)
910 CALL HCHAR(13,15,116,4):: CA
    LL VCHAR(14,18,116,2):: CALL
     VCHAR(15,19,116,3):: CALL V
    CHAR(17,20,116,2)
920 CALL HCHAR(18,22,116,2):: CA
    LL HCHAR(15,23,116,4):: CALL
     HCHAR(12,26,116,4):: CALL V
    CHAR(16,23,116,2)
930 CALL VCHAR(13,26,116,2):: CA
    LL VCHAR(10,29,116,3)
940 CALL HCHAR(24,1,116,32)
950 REM SPRITES
960 CALL SCREEN(5)
970 CALL MAGNIFY(3)
980 CALL SPRITE(#3,132,15,INT(RN
    D*150)+20,INT(RND*220)+20)::
     CALL SPRITE(#7,132,11,INT(R
    ND*150)+20,INT(RND*220)+20)
990 CALL SPRITE(#6,136,15,INT(RN
    D*150)+20,INT(RND*220)+20)::
     CALL SPRITE(#8,136,16,INT(R
    ND*150)+20,INT(RND*220)+20)
1000 CALL SPRITE(#4,140,11,INT(R
    ND*150)+20,INT(RND*220)+20)
    :: CALL SPRITE(#5,140,2,INT
    (RND*150)+20,INT(RND*220)+2
    0)
1010 CALL SPRITE(#9,132,16,INT(R
    ND*150)+20,INT(RND*220)+20)
    :: CALL SPRITE(#10,136,11,I
    NT(RND*150)+20,INT(RND*220)
    +20)
1020 CALL SPRITE(#11,140,15,INT(
    RND*150)+20,INT(RND*220)+20
    )::  CALL SPRITE(#12,108,16,
    INT(RND*150)+20,INT(RND*220
    )+20)
1030 CALL COINC(ALL,FF):: IF FF=
    -1 THEN 980
1040 X=INT(RND*220)+30 :: Y=INT(
    RND*160)+20
1050 CALL SPRITE(#1,96,2,120,100
    )
1060 FOR I=0 TO 30 STEP 5 :: CAL
    L SOUND(-100,215,I,215*2,I,
    215*3,I,-3,I):: NEXT I
1070 CALL SPRITE(#2,100,2,Y,X,0,
    -5)
1080 REM SPIELERSCHLEIFE
1090 CALL KEY(0,KEY,S)
1100 IF S=0 THEN CALL MOTION(#1,
    0,0):: GOTO 1160
```



```
1110 IF KEY=88 THEN CALL
    PATTERN(#1,120):: CALL
    MOTION(#1,7,0):: GOTO 1160
1120 IF KEY=83 THEN CALL PATTERN
    (#1,128):: CALL MOTION(#1,0
    ,-7):: GOTO 1160
1130 IF KEY=69 THEN CALL PATTERN
    (#1,112):: CALL MOTION(#1,-
    7,0):: GOTO 1160
1140 IF KEY=68 THEN CALL PATTERN
    (#1,96):: CALL MOTION(#1,0,
    7):: GOTO 1160
1150 GOTO 1090
1160 CALL POSITION(#1,Y1,X1):: C
    ALL POSITION(#2,Y2,X2)
1170 CALL COINC(ALL,CC):: IF CC=
    -1 THEN GOSUB 1260
1180 IF Y2-Y1>0 THEN 1220
1190 IF X2-X1>0 THEN 1210
1200 CALL PATTERN(#2,104):: CALL
     MOTION(#2,V,V):: GOTO 1090
1210 CALL PATTERN(#2,100):: CALL
     MOTION(#2,V,-V):: GOTO 109
    0
1220 IF X2-X1>0 THEN 1240
1230 CALL PATTERN(#2,104):: CALL
     MOTION(#2,-V,V):: GOTO 109
    0
1240 CALL PATTERN(#2,100):: CALL
     MOTION(#2,-V,-V):: GOTO 10
    90
1250 REM SCHATZ GEHOBEN
1260 CALL COINC(#1,#2,10,DD):: I
    F DD=-1 THEN 1390
1270 CALL SOUND(100,-1,0)
1280 CALL MOTION(#2,0,0)
1290 FOR I=3 TO 12 :: CALL COINC
    (#1,#I,10,EE)
1300 IF EE=-1 THEN CALL DELSPRIT
    E(#I):: GOTO 1330
1310 NEXT I
1320 GOTO 1370
1330 PUNKTE=PUNKTE+100
1340 FOR I=783 TO 523 STEP -260
    :: FOR J=0 TO 30 STEP 8 ::
    CALL SOUND(-100,I,J,I*2,J,I
    *3,J):: NEXT J :: NEXT I
1350 DISPLAY AT(1,2):"TAUCHER:";
    TAUCHER;"   PUNKTE:";PUNKTE
1360 IF PUNKTE/1000=INT(PUNKTE/1
    000)THEN 1530
```



```
1370 RETURN
1380 REM TAUCHER GEFRESSEN
1390 CALL SOUND(200,-7,0)
1400 CALL POSITION(#2,X0,Y0):: I
     F Y0<9 THEN 1420
1410 CALL SPRITE(#20,124,9,X0,Y0
     -10)
1420 FOR I=550 TO 110 STEP -10 :
     : CALL MOTION(#1,3,0,#20,3,
     0):: CALL SOUND(-90,I,0,I*3
     ,0,-3,0):: NEXT I
1430 CALL DELSPRITE(#1)
1440 CALL DELSPRITE(#20)
1450 FOR I=1 TO 5 :: CALL SOUND(
     -200,-7,0):: FOR J=1 TO 50
     :: NEXT J :: NEXT I
1460 CALL DELSPRITE(#2)
1470 TAUCHER=TAUCHER-1
1480 DISPLAY AT(1,2):"TAUCHER:";
     TAUCHER;"   PUNKTE:";PUNKTE
1490 IF TAUCHER=0 THEN 1650
1500 FOR I=1 TO 300 :: NEXT I
1510 GOTO 1040
1520 REM GEWONNEN
1530 CALL DELSPRITE(ALL)
1540 CALL SPRITE(#1,112,2,160,53
     ):: DISPLAY AT(12,8):"SCHAT
     Z GEHOBEN"
1545 V=V+2
1550 FOR I=1 TO 19 :: CALL SOUND
     (T1(I),T2(I),0,T3(I),0,-3,0
     ):: NEXT I
1560 FOR I=1 TO 3
1570 CALL PATTERN(#1,112):: CALL
      MOTION(#1,-20,0)
1580 FOR J=1 TO 400 :: NEXT J ::
      CALL PATTERN(#1,96):: CALL
      MOTION(#1,0,20)
1590 FOR J=1 TO 400 :: NEXT J ::
      CALL PATTERN(#1,120):: CAL
     L MOTION(#1,20,0)
1600 FOR J=1 TO 400 :: NEXT J ::
      CALL PATTERN(#1,128):: CAL
     L MOTION(#1,0,-20)
1610 FOR J=1 TO 400 :: NEXT J ::
      NEXT I
1620 CALL DELSPRITE(ALL)
1630 GOTO 1740
1640 REM VERLOREN
1645 V=0
1650 CALL DELSPRITE(ALL)
1660 CALL MAGNIFY(4)
1670 CALL SPRITE(#1,100,2,90,120
     )
1680 FOR I=1 TO 10
1690 CALL PATTERN(#1,100):: CALL
      MOTION(#1,0,-20)
1700 FOR J=1 TO 250 :: NEXT J
1710 CALL PATTERN(#1,104):: CALL
      MOTION(#1,0,20)
1720 FOR J=1 TO 250 :: NEXT J
1730 NEXT I
1740 DISPLAY AT(24,2):"NEUES SPI
     EL ?  (J/N)"
1750 CALL KEY(0,K,S)
1760 IF K=78 THEN 2050
1770 IF K=74 THEN 1790
1780 GOTO 1750
1790 CALL DELSPRITE(#1)
1800 TAUCHER=5
1810 GOTO 840
1820 REM TITEL
1830 CALL CLEAR
1840 CALL SCREEN(5)
1850 CALL MAGNIFY(4)
1860 CALL SPRITE(#1,128,2,40,1,0
     ,-30,#2,100,2,40,150,0,-35)
1870 DISPLAY AT(12,3):"S C H A T
      Z T A U C H E R "
1880 FOR I=1 TO 300 :: NEXT I
1890 DISPLAY AT(24,2):"REGELN ER
     WUENSCHT ? (J/N)"
1900 CALL KEY(0,K,S)
1910 IF K=78 THEN CALL DELSPRITE
     (ALL):: RETURN
1920 IF K=74 THEN 1950
1930 GOTO 1900
1940 REM SPIELREGELN
1950 CALL CLEAR
1960 CALL MAGNIFY(3)
1970 CALL LOCATE(#1,20,1,#2,20,1
     50)
1980 DISPLAY AT(6,2):"EINE CREW
     VON 5 TAUCHERN    WILL DEN
     SCHATZ DER TITANIC BERGEN."
1990 DISPLAY AT(12,2):"EIN BOESE
     R FISCH WILL SIE   DARAN HI
     NDERN !!"
2000 DISPLAY AT(17,2):"STEUERUNG
      = PFEILTASTEN."
2010 DISPLAY AT(20,2):"VIEL GLUE
     CK !"
2020 INPUT "START MIT ENTER--->"
     :EN$
2030 CALL DELSPRITE(ALL):: RETUR
     N
2040 REM ENDE
2050 CALL CLEAR
2060 CALL DELSPRITE(ALL)
2070 CALL SAY("GOODBYE")
2080 END
```

Listing zu »Schatztaucher« (Schluß)

Grundvoraussetzung für alle drei Programme ist das Betriebssystem OS9. Geliefert werden die Programme von Dragon in einer komfortablen Verpackung. Als Datenträger dient eine Diskette. Ein ausführliches Bedienungshandbuch rundet das jeweilige Paket ab. Zum Arbeiten mit den Programmen sind recht gute Englischkenntnisse notwendig, denn die Handbücher und die eigentlichen Programme sind — wie viele andere professionelle Software auch — in Englisch geschrieben. Wie uns der Importeur mitteilte, wird aber bereits an den deutschen Übersetzungen gearbeitet. So kann auch der Anwender, dessen Fremdsprachenkenntnisse nicht so gut sind, mit für ihn brauchbaren Programmen rechnen. Bei Erscheinen sollen übrigens die deutschen Versionen kostenlos nachgeliefert werden.

**Neben den bereits im Testbericht erwähnten Sprachen Pascal, C und Basic 09 konnten wir noch die drei professionellen Programme »Stylograph« (Textverarbeitung), »RMS« (Datenverwaltung) und »Dynacalc« (Tabellenkalkulation) für Sie testen.**

Das für den Dragon 64 angebotene Textverarbeitungssystem besteht im Grunde aus drei Teilprogrammen, die gemeinsam auf einer Diskette gespeichert sind: Das eigentliche Textverarbeitungsprogramm Stylograph, das dem Bearbeiten von Textdateien dienende Mailmerge und Spellcheck, ein Programm das den Text auf Rechtschreibungsfehler hin überprüfen kann.

Stylograph dient zum Erstellen und Korrigieren von Texten auf dem

# Software für den Dragon 64

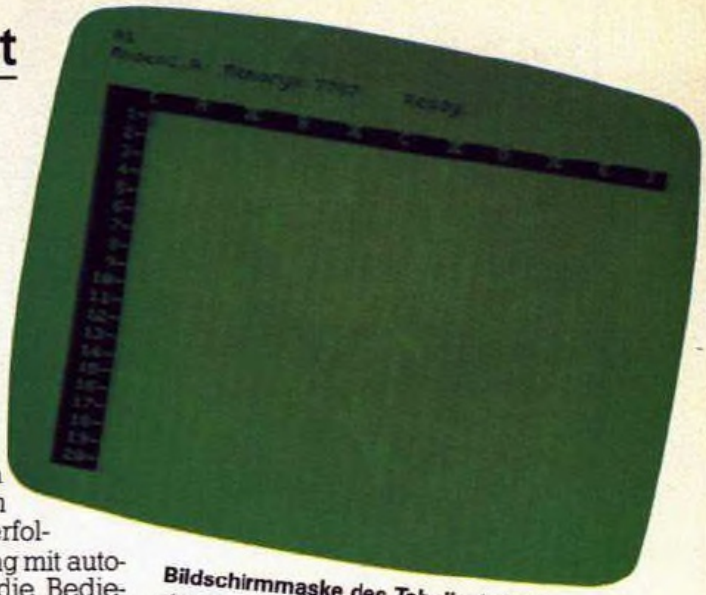
# Geschen

**Geschenke sollten normalerweise Freude bereiten. Wie oft setzt man sich über das »ach, Du brauchst mir nichts zu schenken« hinweg. Vielleicht hätte man es manchmal doch besser beherzigt. Denn welche Tücken — hoffentlich nur im Ausnahmefall — das Geschenk »Computer« haben kann, beschreibt der folgende Bericht.**

Bevor man Computer-Anwender wird, muß man zuerst einmal einsteigen. So schaute ich mir im November letzten Jahres entsprechende Inserate an — was meine Familie veranlaßte, mir einen TI-99/4A mit Teco Monitor und einem Seikosha-Drucker, Modell GP-100A, zu schenken. Sowohl meine Frau als auch meine Tochter wollten mir natürlich damit eine Freude machen. Und wie ich mich freute — das richtige Modell für mich als »Einsteiger«.

Als ich noch am gleichen Tag die Geräte installierte, stellte ich fest, daß der Drucker nicht ging. Nicht weiter tragisch — dachte ich. Ein Anruf bei dem Kaufhaus, in dem die Geräte erstanden worden waren, sollte klären, was zu tun sei, ob ich vielleicht einen Fehler gemacht hätte. Der Verkäufer hatte eine zweifelhaft erfrischende Art mir zu antworten: »Manche Kunden meinen, wenn sie den Namen ihrer Großmutter eingeben, funktioniert alles.« Das wollte ich nicht so einfach hinnehmen und sprach am Tag darauf persönlich mit dem Verkäufer. Er war der Ansicht, jedes Kind (nicht: Jedes computerinteressierte Kind!) wisse, daß der TI-99/4A nicht mehr hergestellt werde. Meine Frau wußte es nicht und ich bin der Ansicht, das wäre auch gar nicht notwendig gewesen. Sie hätte beim Kauf der Geräte darauf hingewiesen werden müssen. Auch aus den Inseraten ging das nicht hervor; es hätte heißen müssen: TI-99/4A Auslaufmodell.

»Das letzte Wort zur Technologie« von Texas Instruments mutet in diesem Zusammenhang auch recht befremdlich an: »Der Homecomputer TI-99/4A ist ein 16-Bit-Computer. Dies bedeutet, daß man sich noch viele Jahre auf die Entwicklung kompatibler Software verlassen kann. Damit sind Sie mit Ihrem Texas Instruments Homecomputer bereits weit voraus.« Vielleicht war ich bereits weit voraus, aber ich konnte es einfach nicht feststellen.



**Bildschirmmaske des Tabellenkalkulationsprogramms Dynacalc**

# k mit Tücken

Bildschirm und anschließender Ausgabe über einen Drucker. Laut Herstellerangabe lassen sich nahezu alle Drucker ansteuern, die über einen parallelen Anschluß verfügen.

Die Ausgabe auf dem Bildschirm erfolgt in der kleinen Darstellung des OS9 wie sie schon im Hardware-Testbericht beschrieben wurde (51 Zeichen pro Zeile und 24 Zeilen pro Bildschirmseite). Durch die gedrängte Darstellung wird das Lesen ziemlich erschwert, besonders wenn man mit einem Fernsehgerät arbeitet. Aber auch für einen Monitor ist das Schriftbild nicht ideal, da das Auflösungsvermögen des Dragon 64 für solche kleinen Buchstaben einfach zu grob ist. Will man mit den großen Buchstaben arbeiten, so steigt das Programm sofort nach dem Laden aus. Die Umschaltung auf die kleinen Buchstaben muß mit dem Befehl »go51« per Hand erfolgen; hier hätte eine Lösung mit automatischer Umschaltung die Bedienerfreundlichkeit erhöht. Überhaupt ist der Programmstart etwas umständlich, denn bevor man OS9-Software laden kann, müssen die Directorys auf die jeweilige Programmdiskette umgeschrieben werden.

Auch mit der Tastenbelegung muß man sich erst anfreunden, da sie etwas ungewohnt ist. So erfolgt

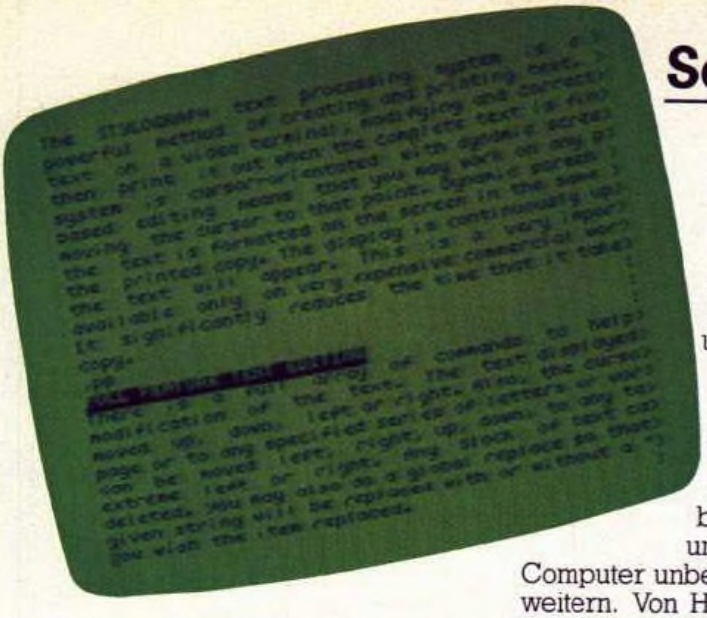
Am nächsten Tag beschwerten wir uns gemeinsam, meine Frau und ich, bei der Geschäftsleitung des Kaufhauses. Wir wurden gebeten, den Drucker zur Überprüfung zu bringen — also einen Tag später nochmals hin. Der zuständige Techniker empfing mich mit den Worten: »Das werden wir gleich haben.« Als nach einer halben Stunde ziemlich deutlich wurde, daß er nicht in der Lage war, das Oberteil abzumontieren, machte ich ihn darauf aufmerksam, daß die Schrauben an der Unterseite des Geräts zu lösen sind. Trotz aller nachfolgenden Bemühungen seitens des Technikers, der Drucker war einfach nicht zum Drucken geeignet. Es wurde ein neuer Drucker geholt — aber er zeigte sich ebenso »störrisch«, es war kein Testausdruck möglich. Ursache: In der deutschen Bedienungsanleitung fehlte ein Hinweis auf die Stellung des Schalters (dip switch) Nummer 4, eindeutig ein Übersetzungsfehler! Erst nachdem ich dem Techniker die amerikanische Ausführung vorgelesen hatte, funktionierte der neue Drucker. Jetzt war es endlich soweit: Nach drei Stunden Aufenthalt konnte ich den neuen Drucker mit nach Hause nehmen.

Nun wollte ich loslegen mit dem Programmieren, da mußte ich zu meinem Schrecken feststellen, daß mir etwas fehlte: »...beginnen Sie am besten mit dem Buch 'Basic für Anfänger', im Lieferumfang enthalten« — so die Einführung der TI-Bedienungsanleitung. Ich forderte das Buch nach, was die Antwort zur Folge hatte, daß man vorhabe, den Inhalt in die Bedienungsanleitung zu integrieren

Als nächstes kam das Fiasko mit dem Kassettenkabel. Mir wurde eine Lernkassette mit dem Hinweis verkauft, daß ein Kabel gar nicht notwendig sei. Erst nach längerem penetrantem Nachforschen und mehreren Ferngesprächen erhielt ich ein Kabel von Gütersloh.

Ich beschloß, weiteres Zubehör nicht mehr in dem Kaufhaus zu erstehen und kaufte eine TI-Programmkassette woanders. Dort wurde mir ausdrücklich versichert, daß dazu außer der Standardausführung des TI-99/4A und dem Monitor nichts erforderlich sei. Zu Hause mußte ich dann im Kleingedruckten in englisch lesen, daß ich noch ein Modul benötigte. Als ich das Modul bestellen wollte, war die Antwort: »Nicht mehr lieferbar«. Die daraufhin zurückgegebene Programmkassette wollte ich dann gegen Leerkassetten eintauschen — damit konnte ich nichts falsch machen. Doch der Verkäufer hatte eine bessere Idee. Er schlug mir vor, eine von den Atari-Spielekassetten zu nehmen, »die ohne weiteres Zubehör auf dem TI laufen«. Ich zahlte 50 Mark drauf und stellte zu Hause fest, daß ich eigentlich einen Joystick dazu benötigte. Die Antwort bei der Bestellung desselben lautete wie vorher: »Leider nicht mehr lieferbar«.

Also wandte ich mich wieder meinem Drucker zu, er sollte ein einfaches Zeichen erzeugen. Keine Grafik, sondern einen durchgehenden Strich oder ein »O«. Es war nicht möglich. Ich wandte mich an Texas Instruments. Dort verwies man mich an Seikosha. In dem Kaufhaus, wo ursprünglich einmal der »Stein des Anstoßes« ins Rollen gekommen war, erfuhr ich dann auch die Adresse von Seikosha. Ich rief in Hamburg an. Dort verwies man mich an einen Programmierer in Weghe. Ich rief in Weghe an. Der Programmierer verwies mich an einen Kontaktmann in meiner Heimatstadt, der noch am selben Tag bei mir vorbeikam. Aber bis heute hat der Drucker noch kein einziges Symbol ausgespuckt. Bemerken möchte ich noch, daß alle beteiligten Personen — ausgenommen der Verkäufer in dem Kaufhaus — sehr höflich und zuvorkommend waren. Es konnte mir aber keiner den Eindruck vermitteln, daß er auch nur die leiseste Ahnung von der Funktion meines Computers und Druckers hat. (A. Graup)



**Textverarbeitungsprogramm Stylograph**

beispielsweise die Cursorsteuerung nicht wie üblich mit den Pfeiltasten, sondern mit Tasten aus dem normalen Tastenfeld, die abhängig vom Modus verschieden belegt sind. So bewirken die Tasten »7« und »9« Sprünge um 25 Spalten nach links und rechts, eine Wahl die sicher nicht besonders einleuchtend ist.

Mit der zur CONTROL-Taste umfunktionierten CLEAR-Taste kann man verschiedene Kommandos und Hilfsroutinen aufrufen. Diese Befehle sind in allen verfügbaren Modi gleich. Neben einer Help-Funktion, die eine Liste aller möglichen Kommandos liefert, existiert eine Unterroutine, die den momentanen Status des zu bearbeitenden Textes anzeigt. Hier findet man auf einem Blick, welche Textdatei gerade bearbeitet wird, welche Voreinstellungen für die Formatierung gewählt wurden und ähnliches. Besonders angenehm ist uns eine Hilfsroutine für Fehlermeldungen aufgefallen. Da Stylograph die Fehler des Benutzers nicht direkt anzeigt, sondern mit einem Piepston quittiert, wird der entsprechende Fehler erst mit dieser Unterroutine dokumentiert.

## Ein alter Bekannter und eine »nützliche« Hilfe: Mailmerge und Spellcheck

Zum Erstellen von Formbriefen oder anderen Texten mit variablen Elementen findet man auf der Diskette das Programm Mailmerge, wie es bei vielen Textverarbeitungssystemen verwendet wird. Ein hoffentlich nicht ganz ernst gemeintes Beispiel ist ein Formbrief an den amerikanischen Präsidenten, der als Variable den Namen des Präsidenten, seine Partei, die Partei des Absenders und einiges mehr vorsieht.

Das ganze Paket der Textverarbeitung wird durch das Unterprogramm Spellcheck abgerundet. Spellcheck ist eine Routine, die eine Textdatei auf eventuelle Rechtschreibfehler hin überprüft und vermeintlich gefundene anzeigt. Nun hat man die Möglichkeit, den Fehler zu korrigieren oder die Datenbank von Spellcheck um ein bis dahin dem Computer unbekanntes Wort zu erweitern. Von Haus aus besitzt das Programm einen Grundwortschatz von 42000 Wörtern. Leider sind auch hier nur englische Wörter erfaßt, so daß bis jetzt ausschließlich englische Texte mit Spellcheck geprüft werden können. Jedoch ist es leicht möglich, eine neue Datei für die deutsche Sprache anzulegen, vorausgesetzt man findet jemanden, der Lust und Laune hat, die Wörter einzutippen. Somit bleibt die Nützlichkeit von Spellcheck, solange keine deutsche Version angeboten wird, doch ein wenig eingeschränkt.

## Datenverarbeitung mit »Record Managment System«

Grundvoraussetzung zum Betrieb dieses Programms zur Datenverwaltung ist neben OS9 ein Textverarbeitungssystem für den Dragon 64. Neben dem von uns getesteten Stylograph wird auch ein Maschinensprachenediter angeboten, der uns aber leider nicht zum Testen vorlag. Mit dem Textverarbeitungssystem muß man die Masken für die Datenfelder erstellen, ohne die das RMS-Programm nicht lauffähig ist. Somit muß auch wieder auf den kleinen Zeichensatz umgestellt werden und der Programmstart gestaltet sich ähnlich wie bereits beschrieben.

Die Dateiverwaltung stellt ein Anwendungspaket dar, mit dem Karteikartensammlungen überflüssig und Kontoführungen vereinfacht werden. Die bearbeiteten Zahlen können mit einem Drucker oder über den Bildschirm in Grafiken veranschaulicht werden. Dazu kann man mit RMS die Daten auch von anderen Sprachen aus aufrufen.

Zur Bearbeitung der Daten kann man verschiedene Felder unterschiedlichster Typen eröffnen. Die Tastatur wird mit Schlüsselwörtern belegt, die wieder mit Hilfe der umfunktionierten CLEAR-Taste aufgerufen werden. Einige wenige Routinen werden in Kombination mit der SHIFT-Taste angesteuert.

Obwohl es sich — wie der Hersteller behauptet — um ein leicht zu beherrschendes Programm handeln soll, ist doch eine nicht geringe Zeit notwendig, um sich in dieses recht komplexe System einzuarbeiten. Dies wird mit ausführlichen Beispielen, sowohl im Handbuch, wie auch auf der Diskette, erleichtert.

Der Start von Dynacalc erfolgt analog zu den anderen Programmen. Wieder müssen der kleine Zeichensatz angewählt und einige Directorys von der OS9-Diskette auf die Programmdiskette übertragen werden. Nun hat man ein Programm für die Bearbeitung von Tabellen, welches mit mehr als 25 mathematischen Funktionen arbeitet. Neben den bekannten trigonometrischen und logarithmischen Funktionen findet man unter anderem Funktionen, um aus Zahlenreihen den minimalen und maximalen Wert zu finden oder um Rundungen unter bestimmten Gesichtspunkten zu vollziehen.

Dynacalc simuliert im Prinzip ein großes Arbeitsblatt mit Zeilen und Spalten. Jedes Feld kann entweder numerische Werte, mathematische Operatoren oder Texte enthalten. Diese Daten in den einzelnen Feldern werden nun arithmetisch oder logisch miteinander verknüpft und bearbeitet. Hierzu verwendet man die Cursortasten und wieder die in der gewohnten Weise umfunktionierte Tastatur. Die Ausgabe der Ergebnisse erfolgt über den Bildschirm oder einen angeschlossenen Drucker. Eine grafische Umsetzung der Zahlenwerte ist zwar möglich, aber leider nicht sehr aussagekräftig. Für die Bildschirmausgabe fanden wir nur die Möglichkeit, die Zahlen als horizontales Balkendiagramm darzustellen, jedoch ist die Auflösung zu grob um differenzierte Ergebnisse zu erhalten.

Trotz einiger nicht besonders gut gelöster Teilprobleme kann man die für ein Tabellenkakulationsprogramm typischen Aufgaben recht gut bewältigen. Ausführliche Beispiele helfen hierbei — wie schon bei den beiden anderen Programmen — den Benutzer schnell und sicher in die Geheimnisse der Tabellenkalkulation einzuführen.

Mit ihren Preisen zwischen 250 und 400 Mark gehören die besprochenen Programme nicht zu den teuersten Angeboten ihrer Kategorie und haben ein akzeptables Preis/Leistungsverhältnis. Sie sind übrigens auch auf dem Dragon 32 mit der 64-KByte-Erweiterung lauffähig. (Wolfgang Czerny/hg)

# Kampf den Minen

## Spielbeschreibung auf Seite 40

| | |
|---|---|
| M(3,3) bis M(21,19) | nimmt einen der Werte 0/1/2/3 an. Die erste Zahl des Arrays gibt die Zeilenposition an, die zweite die Spaltenposition, an der sich keine Mine (0) oder ein Block (1) oder eine hörbare Mine (2) oder eine nicht ortbare Mine (3) befindet |
| X, XX | gibt die Spaltenpositon an, an der sich der Tank befindet |
| Y, YY | gibt die Zeilenpositon an, an der sich der Tank befindet |
| PX | gibt die hochaufgelöste Spaltenpositon zu X an, die für die Sprites benötigt wird |
| PY | gibt die hochaufgelöste Zeilenposition zu Y an, die für die Sprites benötigt wird |
| E | Spaltenposition, an der der Tank ins Spielfeld fährt |
| M2 | Anzahl der hörbaren Minen die zu Spielbeginn gesetzt werden |
| EZ | Zeilenposition, an der sich das unterste der Reserveboote befindet |
| F | gibt die Punktzahl an, die für jedes vorgerückte Feld gegeben wird |
| NR | gibt die Punktzahl an, bei der es neun neue Tanks gibt |
| MH | gibt die Anzahl der sich noch im Spiel befindlichen hörbaren Minen an |
| ML | gibt die Anzahl der sich noch im Spiel befindlichen nicht ortbaren Minen an |
| B | gibt die Bonus-Punktzahl an |
| S | gibt die Score-Punktzahl an |
| H | gibt die Highscore-Punktzahl an |
| ZE | gibt die Zeilenposition an, in die die nächste Mine gesetzt wird |
| SP | gibt die Spaltenpositon an, in die die nächste Mine gesetzt wird |
| ST,K | Steuervariablen für CALL KEY |
| Zeit | Zeitschleifenvariable |
| P | gibt die Spaltenposition an, an der das Schiff steht |
| SN, SN $ | = alter Score + Bonus. Wird benötigt, um beim Übertrag Bonus auf den Score einen mechanischen Zähleffekt zu simulieren |
| I,J | Schleifenvariablen |

**Variablenbeschreibung**

```
100 ! TANK
110 !
120 ! COPYRIGHT MICHAEL DANNER
130 !           SCHEFFELSTR. 14
140 !           7803 GUNDELFINGE
    N
150 !
160 ! DEFINITIONEN
170 !
180 DIM M(23,21)
190 DEF PX=(X-1)*8+1
200 DEF PY=(Y-1)*8+1
210 CALL CLEAR :: CALL SCREEN(2)
    :: CALL MAGNIFY(3):: RANDOMI
    ZE :: E=3 :: M2=6 :: EZ=19 :
    : F=10 :: NR=2000
220 !
230 ! ( BREITER ZEICHENSATZ )
240 !
250 DATA 1818181818001818,636363
    63,42FFFF4242FFFF422,6CFED0F
    C7E16FE6C,C6CE1E3C78F0E6C6,7
    0F8D0767ECCFE76,18182,1C3830
    303030381C
260 DATA 381C0C0C0C0C38,42243CFF
    FF3C2442,0018187E7E1818,0000
    00000018182,0000007E7E,00000
    00000001818,060E1E3C78F0E0C
270 DATA 7EFE838DB1C1FF7E,183878
    7818181818,7CFE0E3C78E0FEFE,
    7CFE0E3C3C0EFE7C,1C3C7CECFEF
    E0C0C,FEFEC0FCFE06FE7C,7CFEC
    0FCFEC6FE7C
280 DATA FEFE060E1C38303,7CFEC67
    C7CC6FE7C,7CFEC6FE7E06FE7C,0
    0181800001818,0018180000181
    2
290 DATA 061E7CF0F07C1E06,007C7C
    00007C7C,C0F07C1E1E7CF0C,7CF
    E061E1C001818,7CFEC6DEDEDEDE
    C
300 DATA 387CEEC6FEFEC6C6,FCFEC6
    FCFCC6FEFC,7CFEC0C0C0C0FE7C,
    FCFEC6C6C6C6FEFC,FEFEC0F8F8C
    0FEFE,FEFEC0F8F8C0C0C0,7CFEC
    0CECEC6FE7C
310 DATA C6C6C6FEFEC6C6C6,181818
    1818181818,FEFE06060606FE7C,
    C6CEDCF8F8DCCEC6,C0C0C0C0C0C
    0FEFE,C6EEFEFED6C6C6C6
320 DATA C6E6E6F6DECECEC6,7CFEC6
    C6C6C6FE7C,FCFEC6FEFCC0C0C0,
    7CFEC6C6D6CCFE76,FCFEC6FEFCD
    CCEC6,7CFEC0FC7E06FE7C
330 DATA FEFE181818181818,C6C6C6
    C6C6C6FE7C,C6C6EE6C7C38381,C
    6C6C6C6D6FEFE6C,C6EE7C38387C
    EEC6,C6CEFC783030303
340 DATA FEFE0E1C3870FEFE,3C3830
    303030383C,406030180C0602,3C
    1C0C0C0C0C1C3C,0808141422224
    141,000000000000FFFF
350 !
360 ! ( UMRANDUNGEN )
370 !
380 DATA FF8080808080808080,FF0101
    0101010101,8080808080808080FF,
    0101010101010101FF,80808080808
    08080,0101010101010101,FF,00
    000000000000FF
```

```
390 !
400 ! ( TANK & SCHIFF )
410 !
420 DATA 00000101030307070F0F1F1
    F3F3F000000008080C0C0E0E0F0F
    0F8F8FCFC,0000000000000000000
    1C0808FF7F3F1F000000000000000
    000030787878FFFEFCF8
430 !
440 !
450 FOR I=33 TO 103 :: READ A$ :
    : CALL CHAR(I,A$):: NEXT I
460 READ A$ :: CALL CHAR(104,A$)
470 READ A$ :: CALL CHAR(108,A$)
480 CALL CHAR(116,"FFFFFFFFFFFFF
    FFF")
490 !
500 !
510 ! SPIELFELD DRUCKEN
520 !
530 CALL CLEAR
540 FOR I=1 TO 14 :: CALL COLOR(
    I,1,1):: NEXT I
550 !
560 ! ( FELDER )
570 !
580 FOR I=3 TO 19 STEP 2
590 FOR J=3 TO 21 STEP 2
600 CALL HCHAR(J,I,96)
610 CALL HCHAR(J,I+1,97)
620 CALL HCHAR(J+1,I,98)
630 CALL HCHAR(J+1,I+1,99)
640 NEXT J
650 NEXT I
660 CALL VCHAR(3,2,101,20):: CAL
    L VCHAR(3,21,100,20):: CALL
    HCHAR(2,3,103,18)
670 !
680 ! ( ANZEIGE )
690 !
700 DISPLAY AT(4,21):"MINEN"
710 DISPLAY AT(9,21):"BONUS"
720 DISPLAY AT(14,21):"SCORE"
730 DISPLAY AT(19,22):"HIGH"
740 CALL VCHAR(3,22,100,20):: CA
    LL VCHAR(3,28,101,20):: CALL
     HCHAR(3,22,102,7)
750 CALL HCHAR(7,22,103,7):: CAL
    L HCHAR(8,22,102,7):: CALL H
    CHAR(12,22,103,7)
760 CALL HCHAR(13,22,102,7):: CA
    LL HCHAR(17,22,103,7):: CALL
     HCHAR(18,22,102,7)
770 CALL HCHAR(22,22,103,7)
780 FOR I=3 TO 18 STEP 5 :: CALL
     HCHAR(I,22,96):: CALL HCHAR
    (I,28,97):: NEXT I
790 FOR I=7 TO 22 STEP 5 :: CALL
     HCHAR(I,22,98):: CALL HCHAR
    (I,28,99):: NEXT I
800 CALL MINEN(MH,ML)
810 CALL BONUS(B)
820 CALL SCORE(S)
830 CALL HIGHSCORE(H)
840 !
850 ! ( TANKVORRAT )
860 !
870 FOR I=3 TO 19 STEP 2
880 CALL HCHAR(I,30,104):: CALL
    HCHAR(I+1,30,105)
890 CALL HCHAR(I,31,106):: CALL
    HCHAR(I+1,31,107)
900 NEXT I
910 !
920 ! ( 6 MINEN HOERBAR UND 6 MI
    NEN SICHTBAR DRUCKEN )
930 !
940 FOR I=1 TO M2
950 GOSUB 2110
960 NEXT I
970 FOR I=1 TO 6
980 GOSUB 2020
990 NEXT I
1000 !
1010 FOR I=1 TO 9 :: CALL COLOR(
     I,16,1):: NEXT I :: CALL CO
     LOR(10,11,1,11,9,1)
1020 !
1030 !
1040 !
1050 ! SPIELBEGINN
1060 !
1070 ! ( SCHIFF DRUCKEN )
1080 !
1090 GOSUB 2290
1100 !
1110 ! ( TANK HOLEN )
1120 !
1130 FOR ZEIT=1 TO 200 :: NEXT Z
     EIT
1140 Y=EZ
1150 CALL SPRITE(#1,104,11,PY,23
     3)
1160 CALL HCHAR(Y,30,32,2):: CAL
     L HCHAR(Y+1,30,32,2)
1170 FOR I=PY TO 177 STEP 8
1180 CALL LOCATE(#1,I,233)
1190 CALL SOUND(-150,(I+100),9)
1200 NEXT I
1210 !
```

**Listing zu «Kampf den Minen«**

```
1220 X=E
1230 FOR I=233 TO PX STEP -8
1240 CALL LOCATE(#1,177,I)
1250 CALL SOUND(-150,(I+100),9)
1260 NEXT I
1270 K=87 :: XX=E :: E=E+2 :: EZ
     =EZ-2 :: YY=23 :: GOTO 1320
1280 !
1290 ! ( TASTATUREINGABE )
1300 !
1310 CALL KEY(0,K,ST)
1320 IF K=87 THEN GOSUB 1540
1330 IF K=65 AND XX<>3 THEN GOSU
     B 1660
1340 IF K=83 AND XX<>19 THEN GOS
     UB 1780
1350 IF K=90 AND YY<>21 THEN GOS
     UB 1900
1360 IF K<>65 AND K<>83 AND K<>8
     7 AND K<>90 THEN 1310
1370 !
1380 ! ( EXPLOSION ? )
1390 !
1400 IF M(YY,XX)<>0 THEN 2530
1410 !
1420 ! ( WARNTON ? )
1430 !
1440 IF M(YY-2,XX)=2 OR M(YY+2,X
     X)=2 OR M(YY,XX+2)=2 OR M(Y
     Y,XX-2)=2 THEN CALL SONAR
1450 !
1460 ! ( OBEN ANGEKOMMEN ? )
1470 !
1480 IF YY=3 THEN 2370
1490 !
1500 GOTO 1310
1510 !
1520 ! OBEN FAHREN
1530 !
1540 X=XX :: Y=YY
1550 FOR I=PY TO PY-16 STEP -4
1560 CALL LOCATE(#1,I,PX)
1570 CALL SOUND(-5,-2,10)
1580 NEXT I
1590 YY=YY-2 :: B=B+F :: CALL BO
     NUS(B)
1600 CALL HCHAR(YY,XX,32,2):: CA
     LL HCHAR(YY+1,XX,32,2)
1610 RETURN
1620 !
1630 !
1640 ! LINKS FAHREN
1650 !
1660 X=XX :: Y=YY
1670 FOR I=PX TO PX-16 STEP -4
1680 CALL LOCATE(#1,PY,I)
1690 CALL SOUND(-5,-2,10)
1700 NEXT I
1710 XX=XX-2
1720 CALL HCHAR(YY,XX,32,2):: CA
     LL HCHAR(YY+1,XX,32,2)
1730 RETURN
1740 !
1750 !
1760 ! RECHTS FAHREN
1770 !
1780 X=XX :: Y=YY
1790 FOR I=PX TO PX+16 STEP 4
1800 CALL LOCATE(#1,PY,I)
1810 CALL SOUND(-5,-2,10)
1820 NEXT I
1830 XX=XX+2
1840 CALL HCHAR(YY,XX,32,2):: CA
     LL HCHAR(YY+1,XX,32,2)
1850 RETURN
1860 !
1870 !
1880 ! UNTEN FAHREN
1890 !
1900 X=XX :: Y=YY
1910 FOR I=PY TO PY+16 STEP 4
1920 CALL LOCATE(#1,I,PX)
1930 CALL SOUND(-5,-2,10)
1940 NEXT I
1950 YY=YY+2 :: B=B-F :: CALL BO
     NUS(B)
1960 CALL HCHAR(YY,XX,32,2):: CA
     LL HCHAR(YY+1,XX,32,2)
1970 RETURN
1980 !
1990 !
2000 ! BLOECKE SETZEN
2010 !
2020 ZE=INT(RND/.1)*2+3
2030 SP=INT(RND/.112)*2+3
2040 IF M(ZE,SP)<>0 THEN 2020
2050 M(ZE,SP)=1 :: CALL HCHAR(ZE
     ,SP,116,2):: CALL HCHAR(ZE+
     1,SP,116,2)
2060 RETURN
2070 !
2080 !
2090 ! HOERBARE MINEN SETZEN
2100 !
2110 ZE=INT(RND/.1)*2+3
2120 SP=INT(RND/.112)*2+3
2130 IF M(ZE,SP)<>0 THEN 2110
2140 M(ZE,SP)=2 :: MH=MH+1 :: CA
     LL MINEN(MH,ML)
2150 RETURN
```

**Listing zu »Kampf den Minen« (Fortsetzung)**

```
2160 !
2170 !
2180 ! LAUTLOSE MINEN SETZEN
2190 !
2200 ZE=INT(RND/.1)*2+3
2210 SP=INT(RND/.112)*2+3
2220 IF M(ZE,SP)<>0 THEN 2200
2230 M(ZE,SP)=3 :: ML=ML+1 :: CA
     LL MINEN(MH,ML)
2240 RETURN
2250 !
2260 !
2270 ! SCHIFF SETZEN
2280 !
2290 X=INT(RND/.112)*2+3 :: P=X
2300 CALL DELSPRITE(#2)
2310 CALL SPRITE(#2,108,5,1,PX)
2320 RETURN
2330 !
2340 !
2350 ! OBEN ANGEKOMMEN
2360 !
2370 CALL SOUND(200,262,2)
2380 FOR ZEIT=1 TO 55 :: NEXT ZE
     IT
2390 CALL SOUND(200,330,2)
2400 FOR ZEIT=1 TO 55 :: NEXT ZE
     IT
2410 CALL SOUND(200,392,2)
2420 FOR ZEIT=1 TO 55 :: NEXT ZE
     IT
2430 CALL SOUND(350,330,2,392,2,
     523,2)
2440 !
2450 GOSUB 2200
2460 IF XX=P THEN B=B+400 ELSE B
     =B+200
2470 CALL MINEN(MH,ML):: CALL BO
     NUS(B)
2480 CALL DELSPRITE(#1)
2490 GOTO 2700
2500 !
2510 ! EXPLOSION
2520 !
2530 CALL COLOR(#1,9):: CALL SOU
     ND(-5,-1,0)
2540 FOR ZEIT=1 TO 12 :: NEXT ZE
     IT
2550 CALL COLOR(#1,11):: CALL SO
     UND(-5,-1,0)
2560 FOR ZEIT=1 TO 12 :: NEXT ZE
     IT
2570 CALL COLOR(#1,9):: CALL SOU
     ND(-5,-1,0)
2580 FOR ZEIT=1 TO 12 :: NEXT ZE
     IT
2590 CALL SOUND(300,-5,0)
2600 CALL DELSPRITE(#1)
2610 CALL HCHAR(YY,XX,116,2):: C
     ALL HCHAR(YY+1,XX,116,2)
2620 IF M(YY,XX)=2 THEN MH=MH-1
2630 IF M(YY,XX)=3 THEN ML=ML-1
2640 CALL MINEN(MH,ML):: M(YY,XX
     )=1
2650 B=B-300 :: CALL BONUS(B)
2660 GOTO 2700
2670 !
2680 ! SCORE
2690 !
2700 FOR ZEIT=1 TO 200 :: NEXT Z
     EIT
2710 !
2720 ! ( SCOREUEBERTRAG )
2730 !
2740 SN=B+S
2750 SN$=STR$(SN):: SN$=RPT$("0"
     ,5-LEN(SN$))&SN$
2760 S$=STR$(S):: S$=RPT$("0",5-
     LEN(S$))&S$
2770 IF SEG$(S$,5,1)<>SEG$(SN$,5
     ,1)THEN S=S+5 :: B=B-5 :: C
     ALL BONUS(B):: CALL SOUND(-
     5,-2,0):: CALL SCORE(S):: G
     OTO 2760
2780 IF SEG$(S$,4,1)<>SEG$(SN$,4
     ,1)THEN S=S+10 :: B=B-10 ::
      CALL BONUS(B):: CALL SOUND
     (-5,-2,0):: CALL SCORE(S)::
      GOTO 2760
2790 IF SEG$(S$,3,1)<>SEG$(SN$,3
     ,1)THEN S=S+100 :: B=B-100
     :: CALL BONUS(B):: CALL SOU
     ND(-5,-2,0):: CALL SCORE(S)
     :: GOTO 2760
2800 IF SEG$(S$,2,1)<>SEG$(SN$,2
     ,1)THEN S=S+1000 :: B=B-100
     0 :: CALL BONUS(B):: CALL S
     OUND(-5,-2,0):: CALL SCORE(
     S):: GOTO 2760
2810 !
2820 !
2830 GOSUB 2290
2840 !
2850 ! ( NOCH TANKS ZUR VERFUEGU
     NG ? )
2860 !
2870 IF EZ<>1 THEN 1130
2880 !
2890 ! ( 9 NEUE TANKS ? ODER SPI
     ELENDE ? )
```

```
2900 !
2910 IF S<NR THEN 3190
2920 !
2930 ! ( 9 NEUE TANKS )
2940 !
2950 FOR I=3 TO 21 STEP 2 :: FOR
      J=3 TO 19 STEP 2 :: M(I,J)
     =0 :: NEXT J :: NEXT I :: M
     H,ML=0
2960 FOR I=3 TO 19 STEP 2
2970 FOR J=3 TO 21 STEP 2
2980 CALL HCHAR(J,I,96)
2990 CALL HCHAR(J,I+1,97)
3000 CALL HCHAR(J+1,I,98)
3010 CALL HCHAR(J+1,I+1,99)
3020 NEXT J
3030 NEXT I
3040 FOR I=1 TO 6
3050 GOSUB 2020
3060 NEXT I
3070 M2=M2+2
3080 FOR I=1 TO M2
3090 GOSUB 2110
3100 NEXT I
3110 FOR I=3 TO 19 STEP 2
3120 CALL HCHAR(I,30,104):: CALL
      HCHAR(I+1,30,105)
3130 CALL HCHAR(I,31,106):: CALL
      HCHAR(I+1,31,107)
3140 NEXT I
3150 E=3 :: EZ=19 :: F=F+5 :: NR
     =NR+3000 :: GOTO 1130
3160 !
3170 !   SPIELENDE
3180 !
3190 IF S<=H THEN 3330
3200 !
3210 ! ( HIGHSCORE )
3220 !
3230 H=S
3240 FOR I=1 TO 3
3250 DISPLAY AT(21,21)SIZE(5):""
3260 CALL SOUND(300,330,2,392,2,
     523,2)
3270 CALL HIGHSCORE(H)
3280 FOR ZEIT=1 TO 60 :: NEXT ZE
     IT
3290 NEXT I
3300 !
3310 ! ( GAME OVER )
3320 !
3330 DISPLAY AT(24,10):"GAME OVE
     R"
3340 FOR ZEIT=1 TO 1000 :: NEXT
     ZEIT
3350 DISPLAY AT(24,3):""
3360 FOR ZEIT=1 TO 20 :: NEXT ZE
     IT
3370 DISPLAY AT(24,3):"PRESS ANY
      KEY TO START"
3380 CALL KEY(0,K,ST):: IF ST<>0
      THEN 3420 ELSE 3380
3390 !
3400 ! ( NEUES SPIEL )
3410 !
3420 MH,ML,B,S=0
3430 M2=6 :: E=3 :: EZ=19 :: F=1
     0 :: NR=2000 :: CALL DELSPR
     ITE(#2):: CALL CLEAR
3440 FOR I=3 TO 21 STEP 2 :: FOR
      J=3 TO 19 STEP 2 :: M(I,J)
     =0 :: NEXT J :: NEXT I
3450 GOTO 530
3460 !
3470 !
3480 !
3490 !
3500 !
3510 SUB MINEN(MH,ML)
3520 DISPLAY AT(6,21)SIZE(2):USI
     NG "##":MH
3530 DISPLAY AT(6,25)SIZE(1):USI
     NG "#":ML
3540 SUBEND
3550 !
3560 SUB BONUS(B)
3570 IF B<0 THEN B=0
3580 DISPLAY AT(11,21)SIZE(5):US
     ING "#####":B
3590 SUBEND
3600 !
3610 SUB SCORE(S)
3620 DISPLAY AT(16,21)SIZE(5):US
     ING "#####":S
3630 SUBEND
3640 !
3650 SUB HIGHSCORE(H)
3660 DISPLAY AT(21,21)SIZE(5):US
     ING "#####":H
3670 SUBEND
3680 !
3690 !
3700 SUB SONAR
3710 !
3720 FOR I=1 TO 3
3730 CALL SOUND(300,400,0)
3740 FOR ZEIT=1 TO 90 :: NEXT ZE
     IT
3750 NEXT I
3760 SUBEND
```

**Listing zu »Kampf den Minen« (Schluß)**

### Listing 1.
### Basic-Lader mit Relokator der RAM-Disk für Programme

```
1 'RAM-DISK für das aktuelle Programm
2 '
3 'Ein Programm kann mit CTRL-S im RAM gesichert und
4 'mit CTRL-L wieder geladen werden. Der Anwender kann die
5 'Puffergröße wählen.
6 '
7 '(C) 1984 Peter Wollschlaeger, Hildesheim
8 '
10 CLS:INPUT"Größe des Puffers in Bytes ";BZ
20 PRINT:PRINT"Moment bitte.....":PRINT
30 HP=16561:HM=PEEK(HP)+256*PEEK(HP+1):'get HIMEM
40 HM=HM-BZ-185:H=INT(HM/256):L=HM-256*H:POKE HP,L:POKE HP+1,H
50 IF HM>32767 THEN HM=HM-65536
60 OG=-4096:'Auf F000H wurde assembliert
70 OS=HM-OG:'Offset
80 FOR I=0 TO 182:READ B:POKE HM+I,B:NEXT
90 FOR I=1 TO 12:READ B
100 AD=HM+B:NA=PEEK(AD)+256*PEEK(AD+1)+OS:H1=INT(NA/256):L1=NA-256*H1
110 POKE HM+B,L1:POKE HM+B+1,H1
120 NEXT
130 H1=INT(BZ/256):L1=BZ-256*H1:POKE HM+181,L1:POKE HM+182,H1
140 DEFUSR=HM:'********** Nur Disk-Basic **************
150 POKE 16526,L:POKE 16527,H:'******** Nur Level 2 ************
160 X=USR(0)
170 PRINT"Pgm-Buffer jetzt bereit"
180 CLEAR 50
190 PRINT"Basic stehen noch ";MEM;" Bytes zur Verfügung"
200 PRINT:PRINT:INPUT"<ENTER> drücken";A$
210 NEW:'******* Hier END einsetzen bis Pgm gesichert und OK *******
220 DATA 42,22,64,34,14,240,33,13,240,34,22,64,201,205,0,0
230 DATA 183,200,254,19,40,35,254,12,192,237,75,179,240,120,177,32
240 DATA 5,33,149,240,24,78,33,195,240,237,91,164,64,237,176,42
250 DATA 177,240,34,249,64,175,195,46,43,42,249,64,34,177,240,237
260 DATA 91,164,64,213,19,19,123,189,32,10,122,188,32,6,209,33
270 DATA 133,240,24,32,209,167,237,82,237,91,181,240,223,56,5,33
280 DATA 163,240,24,16,34,179,240,68,77,42,164,64,17,195,240,237
290 DATA 176,33,122,240,205,117,43,195,204,6,80,71,77,32,83,65
300 DATA 86,69,68,13,0,78,79,32,80,71,77,32,84,79,32,83
310 DATA 65,86,69,13,0,78,79,32,80,71,77,32,83,65,86,69
320 DATA 68,13,0,80,71,77,32,84,79,32,66,73,71,13,0,255
330 DATA 0,255,0,0,0,10,0,4,7,27,34,39,48,61,80,90
340 DATA 101,109,114
```

### Listing 2. Assemblerlisting der »Pgm-Disk«, schlägt im Tempo jede echte RAM-Disk

```
F050            00061 RELA8   EQU   $-2
F052 1820       00062         JR    PRINT          ;NO PGM TO SAVE
F054 D1         00063 PGM     POP   DE
F055 A7         00064         AND   A
F056 ED52       00065         SBC   HL,DE          ;ELSE GET LEN
F058 ED5B85F0   00066         LD    DE,(BUFSIZ)
F05A            00067 RELA9   EQU   $-2
F05C DF         00068         RST   18H            ;COMPARE TO BUFF SIZE
F05D 3805       00069         JR    C,OK
F05F 21A3F0     00070         LD    HL,MSG4        ;IF BAD
F062 1810       00071         JR    PRINT          ;TELL "PGM TO BIG"
F064 22B3F0     00072 OK      LD    (LEN),HL       ;IT FITS, SAVE LEN
F065            00073 RELA10  EQU   $-2
F067 44         00074         LD    B,H            ;BC=LEN
F068 4D         00075         LD    C,L
F069 2AA440     00076         LD    HL,(40A4H)     ;SOURCE
F06C 11C3F0     00077         LD    DE,BUFFER      ;DEST.
F06D            00078 RELA11  EQU   $-2
F06F EDB0       00079         LDIR
F071 217AF0     00080         LD    HL,MSG1        ;TELL
F072            00081 RELA12  EQU   $-2
F074 CD752B     00082 PRINT   CALL  PRTMSG         ;PGM SAVED
F077 C3CC06     00083         JP    READY
F07A 50         00084 MSG1    DEFM  'PGM SAVED'
F083 0D         00085         DEFB  0DH
F084 00         00086         DEFB  0
F085 4E         00087 MSG2    DEFM  'NO PGM TO SAVE'
F093 0D         00088         DEFB  0DH
F094 00         00089         DEFB  0
F095 4E         00090 MSG3    DEFM  'NO PGM SAVED'
F0A1 0D         00091         DEFB  0DH
F0A2 00         00092         DEFB  0
F0A3 50         00093 MSG4    DEFM  'PGM TO BIG'
F0AD 0D         00094         DEFB  0DH
F0AE 00         00095         DEFB  0
0002            00096 HEDSAV  DEFS  2
0002            00097 TALSAV  DEFS  2
F0B3 0000       00098 LEN     DEFW  0000H
F0B5 0A00       00099 BUFSIZ  DEFW  10             ;CHANGED BY LOADER
F0B7 04         00100 RELTAB  DEFB  RELA1-OR
F0B8 07         00101         DEFB  RELA2-OR
F0B9 1B         00102         DEFB  RELA3-OR
F0BA 22         00103         DEFB  RELA4-OR
F0BB 27         00104         DEFB  RELA5-OR
F0BC 30         00105         DEFB  RELA6-OR
F0BD 3D         00106         DEFB  RELA7-OR
F0BE 50         00107         DEFB  RELA8-OR
F0BF 5A         00108         DEFB  RELA9-OR
F0C0 65         00109         DEFB  RELA10-OR
F0C1 6D         00110         DEFB  RELA11-OR
F0C2 72         00111         DEFB  RELA12-OR
F0C3            00112 BUFFER  EQU   $
0000            00113         END
```

# POKE-Listengenerator

**Die Weitergabe von Maschinenprogrammen, die am TI 99/4A mit dem Minimem geschrieben wurden, was bisher auf direkte Programmüberspielungen oder selbstgeschriebene Listings beschränkt. Unser Programm für den TI 99/4A mit Minimem oder Assembler schafft Abhilfe.**

Basic-POKE-Listen bilden auch für Programmanwender ohne Assemblerkenntnisse den Zugang zu Maschinenprogrammen, wenngleich sowohl ihre Erstellung wie auch die Eingabe mühsam und fehlerträchtig ist. Das vorliegende Programm nimmt nun die Mühe der POKE-Listenerstellung ab, indem mit einem einfachen Programmaufruf nach Angabe der Start- und Endadresse die zugehörige Basic-POKE-Liste erstellt wird. Was dann noch zu tun bleibt, ist die Ergänzung der Pokeliste durch eine FOR-NEXT-Schleife mit den POKE-Befehlen für die Zieladresse. Um noch einige interne Pointer zu setzen, die für einen reibungslosen Programmablauf notwendig sind, muß man die fertig erstellte Pokeliste abspeichern, in XBasic einlesen und neuerlich im MERGE-Format abspeichern. Nach dem EinMERGEn in die FOR-NEXT-Schleife verhält sich das Programm so, als würde es über die Tastatur eingegeben werden.

Im Falle der Anwendung des Programms zusammen mit dem Editor-Assembler ist die Absolutstartadresse mit »7118« auf RORG abzuändern, da der vorgegebene Speicherbereich mit dem Editor/Assembler nicht zur Verfügung steht. Auch die Übertragung in ein unter XBasic laufendes Programm ist möglich. Dazu sind die unter REF definierten Utility-Eqautes anzupassen und »XMLINK« (im XBasic-Modul fehlerhaft) durch eine korrigierte Routine analog der Minimem- oder Assemblerversion zu überschreiben.

Das Programm startet mit dem Aufruf: »CALL LINK("PEEK",Startadresse, Endadresse)«. Es generiert ein Basicprogramm von DATA-Zeilen ab Zeile 10 in Zehnerschritten aufwärts.

(Karl Hagenbuchner/lg)

```
POKELISTENGENERATOR auf sich selbst angewendet (>71A8 ->747E)
                                                  29096 - 29822
10 DATA 0,48,1,2,0,10,0,65,147,179,200,4,3,5,42,32
20 DATA 66,65,83,73,67,
30 DATA 32,69,82,83,84,69,76,76,84,32,42,66,89,84,69,83
40 DATA 32,86,69,82,65,82,66,69,73,84,69,84,113,232,114,8
50 DATA 0,0,113,134,0,0,48,4,0,0,48,48,56,53,0,0
60 DATA 0,0,0,0,0,0,0,0,0,0,113,124,114,230,132,0
70 DATA 192,109,0,2,4,195,2,0,0,4,192,129,4,193,60,96
80 DATA 113,172,10,194,9,67,160,194,6,0,22,247,193,131,10,198
90 DATA 9,70,177,160,113,169,6,198,193,67,6,197,9,69,177,96
100 DATA 113,169,177,133,193,67,10,69,9,69,177,96,113,169,6,197
110 DATA 9,67,176,224,113,169,209,67,192,77,2,33,0,6,220,96
120 DATA 113,179,220,69,6,197,220,69,220,70,6,198,212,70,3,128
130 DATA 2,224,113,124,3,0,0,0,4,192,2,2,3,0,2,1
140 DATA 128,0,4,32,96,36,5,128,6,2,22,251,4,192,2,1
150 DATA 0,1,4,32,96,68,4,32,96,28,18,0,200,32,131,74
160 DATA 113,156,5,129,4,32,96,68,4,32,96,28,18,0,200,32
170 DATA 131,74,113,158,104,32,113,156,131,74,200,32,131,74,113,160
180 DATA 200,32,131,74,113,166,4,224,113,164,200,32,131,112,113,162
190 DATA 2,8,113,24,222,32,113,171,222,32,113,176,2,10,0,16
200 DATA 194,96,113,156,4,193,208,121,6,193,4,32,113,228,6,195
210 DATA 208,224,113,178,6,3,145,32,113,169,19,11,222,3,6,195
220 DATA 222,3,222,4,6,196,222,4,222,5,184,32,113,181,113,24
230 DATA 16,25,6,3,209,5,6,196,145,32,113,169,19,10,222,3
240 DATA 6,195,222,3,222,4,6,196,222,4,184,32,113,179,113,24
```

```
250 DATA 16,9,6,3,6,196,222,3,6,195,222,3,222,4,184,32
260 DATA 113,180,113,24,6,32,113,160,19,8,6,10,19,8,222,32
270 DATA 113,177,184,32,113,170,113,24,16,197,7,32,113,164,214,32
280 DATA 113,168,192,32,113,162,208,160,113,24,9,130,5,130,96,2
290 DATA 200,0,113,162,2,3,113,24,208,115,4,32,96,36,5,128
300 DATA 6,2,22,250,2,8,0,16,168,8,113,156,200,32,113,164
310 DATA 113,164,19,158,6,32,113,162,200,32,113,162,131,50,6,32
320 DATA 113,162,192,32,131,112,4,197,6,0,4,32,96,44,144,96
330 DATA 113,176,19,4,128,32,113,162,19,26,16,246,200,0,113,24
340 DATA 2,2,0,2,2,1,113,24,192,32,113,162,4,32,96,40
350 DATA 2,37,0,10,200,5,113,26,5,193,6,64,4,32,96,40
360 DATA 200,0,113,162,6,96,113,162,192,32,113,24,16,227,192,32
370 DATA 113,162,5,192,200,0,131,48,6,64,2,1,113,174,2,2
380 DATA 0,2,4,32,96,40,4,224,131,62,2,0,2,130,2,1
390 DATA 113,182,2,2,0,29,2,7,96,0,188,71,6,2,22,253
400 DATA 2,2,0,29,2,1,113,182,4,32,96,40,124,71,6,2
410 DATA 22,253,192,96,113,166,4,32,113,228,2,1,113,131,208,131
420 DATA 9,130,194,2,188,71,6,8,22,253,2,1,113,131,2,32
430 DATA 0,64,4,32,96,40,2,32,0,7,2,1,113,211,2,2
440 DATA 0,17,192,194,193,3,188,71,6,3,22,253,2,1,113,211
450 DATA 4,32,96,40,124,71,6,4,22,253,4,224,131,124,2,224
460 DATA 131,224,4,96,0,112
```

```
        * BASIC-DATA-LISTE ERSTELLT *

        10 DATA 0,4,192,129,4,193,60
        ,96,113,172,10,194,9,67,16
        20 DATA 6,0,22,247,193,131,1
        0,198,9,70,177,160,113,169,6
        ,198
        30 DATA 193,67,6,197,9,69,17
        7,96,113,169,177,133,193,67,
        10,69
        40 DATA 9,69,177,96,113,169,
        6,197,9,67,176,224,113,169,2
        09,67
        50 DATA 192,77,2,33,0,6,220,
        96,113,179,220,69,6,197,220,
        69
        60 DATA 220,70,6,198,212,70,
        3,128,2,224,113,124,3,0,0,0
        70 DATA 4,192,2,2
       >CALL LINK("DUMP")
```

```
*****************************************************
*                                                   *
* POKELISTEN-GENERATOR FÜR MINIMEM UND EDITOR/ASSEMBLER *
*                                                   *
*      Konfiguration: TI 99/4A + obige Module       *
*                                                   *
* Programmaufgabe: Erstellung von Pokelisten aus einem  *
*                  im RAM befindlichen Maschinenprogramm *
*                  in Form von DATA-Zeilen eines    *
*                  Basicprogramms.                  *
*                                                   *
* Aufruf im Command-Modus: CALL LINK("PEEK",AORG,FFAM)  *
*                                                   *
* (c) Mag.Karl Hagenbuchner              1984/3     *
*                                                   *
*****************************************************
       DEF PEEK

       REF NUMREF,XMLLNK,VSBR,VMBW,VSBW

       AORG >7110          Startadresse für Minimem, sonst RORG

LINE   BSS 100             Puffer für DATA-Zeile mit maximal 16 DATAS
POKER  BSS 32              Registersatz für Maschinenprogramm
AORG   BSS  2              Puffer für Startadresse
FFAM   BSS  2              Puffer für Programmende
NUMBER BSS  2              Puffer für Byteanzahl
TOP    BSS  2              Puffer für aktuelle Programmuntergrenze
RDFLG  BSS  2              Puffer für READY-Flag
BYTES  BSS  2              Puffer für Byteanzahl

ZERO   BYTE 0              DATA-Zeilen-End-Byte
NULL   BYTE '0'            ASCII-Wert von Null
ONE    BYTE 1              Additionsfaktor Zeilenlänge
TWO    BYTE 2              Additionsfaktor für Komma+Zahlenpointer
ZEHN   DATA 10             Divisor in BCD-Routine
ENDE   DATA 65             Pointer vor Linenumber-Table
DATA   BYTE 147            Token für DATA
KOMMA  BYTE 179            Token für ","
ZAHL   BYTE 200            Zahlenpointer-Token
LEN    BYTE 4              Standardlänge BCD-String
DREI   BYTE 3              Grundlänge DATA-Zeile

FIVE   BYTE 5              Längste vorkommende Zahlenlänge inkl.Pointer
       EVEN
TEXT1  TEXT '* BASIC-DATA-LISTE ERSTELLT *'
TEXT2  TEXT 'BYTES VERARBEITET'
       EVEN

CFI    EQU  >1200          Conversion Floating-point to Integer
NEXT   EQU  >0070          Basicreturn
FAC    EQU  >834A          Floating-point-Akkumulator
STATUS EQU  >837C          Adresse Systemstatus
GPLWS  EQU  >83E0          GPL-Workspaceadresse
RAMTOP EQU  >8370          höchste verfügbare VDP-RAM-Adresse
NUMSTA EQU  >8330          Beginn der Zeilennummerntafel
NUMEND EQU  >8332          Ende der Zeilennummerntafel
VARLIS EQU  >833E          Variablenlistenpointer
*****************************************************
*                                                   *
* Subprogramm BCD: wandelt Hexzahl in einen BCD-String  *
*                                                   *
* Eingabe via R1, Ausgabe ab R3: Lengthbyte,w,x,y,z     *
*                                mit führenden Nullen   *
*                                                   *
* Aufruf mit BLWP $BCDLNK                           *
*                                                   *
*****************************************************
BCDLNK DATA BCDWS,BCDGO    BCDLNK-Vektor

BCDWS  BSS 32              Registersatz der Subroutine

BCDGO  MOV @2(R13),R1      Übernahme von R1 des Caller-Programms

CONV   CLR R3              BCD-Puffer löschen
       LI R0,4             4 Stellen sind zu bearbeiten
LOOP   MOV R1,R2           HEX-Zahl kopieren
       CLR R1              Division vorbereiten
       DIV @ZEHN,R1        32-Bit/Hexzahl : 10
       SLA R2,12           Rest ins vordere Byte
       SRL R3,4            BCD-Zahl um eine Stelle nach rechts
       A R2,R3             neuen BCD-Teil addieren
       DEC R0              alle 4 Stellen abzählen
       JNE LOOP            bis alle berücksichtigt

       MOV R3,R6           BCD-Zahl kopieren: 'T/H/Z/E'
       SLA R6,12           Einer separieren
       SRL R6,4            und ins vordere Byte
       AB @NULL,R6         ASCII-Code der Zahl aufbauen
       SWPB R6             ins Lowbyte schieben
       MOV R3,R5           BCD-Zahl kopieren
       SWPB R5             'Z/E' nach vorne
       SRL R5,4            'O/Z' bilden
       AB @NULL,R5         ASCII-Code der Zehnerstelle bilden
       AB R5,R6            und vor ASCII der Einerstelle setzen
       MOV R3,R5           BCD-Zahl kopieren
       SLA R5,4            'T/H' von 'T' befreien
       SRL R5,4            'O,H' bilden
       AB @NULL,R5         ASCII-Code der Hunderterstelle bilden
       SWPB R5             und ins Lowbyte schieben
       SRL R3,4            'O/T'  im Highbyte bilden
       AB @NULL,R3         und ASCII-Code der Tausendestelle machen
       MOVB R3,R5          Tausender vor Hunderter stellen

       MOV R13,R1          Adresse WS der Calling-Routine
       AI R1,6             Pointer auf Adresse von R3
       MOVB @LEN,*R1+      >04 als Lengthbyte vor BCD-String
       MOVB R5,*R1+        Tausenderbyte kopieren
       SWPB R5             Hunderter in Highbyte
       MOVB R5,*R1+        und in String einschieben
       MOVB R6,*R1+        Zehnerbyte in String
       SWPB R6             Einer in Highbyte
       MOVB R6,*R1         und in String einbauen

       RTWP                Rückkehr ins Hauptprogramm
*****************************************************
*                                                   *
*  Beginn des Hauptprogramms PEEK                   *
*                                                   *
*****************************************************
PEEK   LWPI POKER          Laden des eigenen Registersatzes
       LIMI 0              Interrupt unterbinden
       CLR R0              Bildschirmbasisadresse
       LI R2,768           Anzahl zu löschender Bildelemente
       LI R1,>2000         Basic-Blank
CLEAR  BLWP @VSBW          auf Bildschirm schreiben
       INC R0              nächster Bildschirmpunkt
       DEC R2              Anzahl herunterzählen
       JNE CLEAR           bis ganzer Schirm gelöscht
       CLR 0               NUMREF-Vorbereitung
       LI R1,1             Pointer auf erste Variable
       BLWP @NUMREF        AORG-Wert übernehmen
       BLWP @XMLLNK        und seinen Integerwert bilden
       DATA CFI
       MOV @FAC,@AORG      Startadresse speichern
       INC R1              Pointer auf Endadresse
       BLWP @NUMREF        FFAM übernehmen
       BLWP @XMLLNK        und Integerwert bilden
       DATA CFI
       MOV @FAC,@FFAM      FFAM kopieren
       S @AORG,@FAC        Anzahl zu "peekender" Bytes
       MOV @FAC,@NUMBER    in Puffer speichern
       MOV @FAC,@BYTES     Byteanzahl für Ausdruck speichern
       CLR @RDFLG          Ready-Flag löschen
       MOV @RAMTOP,@TOP    Obergrenze des VDPs kopieren

ALOOP  LI 8,LINE           Pufferadresse der DATA-Zeile
       MOVB @TWO,*R8+      Grundlänge voranstellen
       MOVB @DATA,*R8+     Token für DATA
       LI R10,16           16 Bytes in eine DATA-Zeile
       MOV @AORG,R9        diese Bytes ab dieser Adresse bearbeiten

BLOOP  CLR R1              Puffer für gelesenes Byte
       MOVB *R9+,R1        Programmbyte kopieren
       SWPB R1             und in Lowbyte schieben
       BLWP @BCDLNK        BCD-String bilden
       SWPB R3             Lengthbyte ins Lowbyte schieben
       MOVB @ZAHL,R3       Token für Zahlenpointer voranstellen
       DEC R3              Länge von Standard '4' auf 3-stellige Zahl
       CB @NULL,R4         gibt es eine Hunderterstelle ?
       JEQ KEINH           Hunderter nicht vorhanden, nur führende Null
       MOVB R3,*R8+        Zahlenpointer ins Pufferzeile
       SWPB R3             Längenpointer in Highbyte
       MOVB R3,*R8+        und in Zeilenpuffer kopieren
       MOVB R4,*R8+        Hunderterstelle in Puffer
       SWPB R4             Zehner in Highbyte
       MOVB R4,*R8+        und in Puffer kopieren
       MOVB R5,*R8+        Einer in Puffer
       AB @FIVE,@LINE      Gesamtlänge anpassen
       JMP BREADY

KEINH  DEC R3              Längenzähler auf zweistellig korrigieren
       MOVB R5,R4          Einer an die Stelle des fehlenden Hunderter
       SWPB R4             Zahl korrigieren
       CB @NULL,R4         Zehnerstelle <> führende Null ?
       JEQ KEINZ           kein Zehner vorhanden --> Sprung
       MOVB R3,*R8+        Zahlenpointer in Puffer
       SWPB R3             Zahlenlänge nach vorne
       MOVB R3,*R8+        und in Puffer kopieren
       MOVB R4,*R8+        Zehner kopieren
       SWPB R4             Einer ins Highbyte
       MOVB R4,*R8+        und in Puffer kopieren
       AB @LEN,@LINE       Gesamtpufferlänge um 4 erhöhen
```

## TI 99/4A — Spezialitäten

*1. Wie kann man beim TI99/4A mit Extended Basic ein Programm mit Namen und Titel abspeichern, damit durch den Aufruf »OLD CS1. Name« oder »RUN "CS1.Name'« dieses Programm (und nur dieses) geladen und gestartet wird, wie das ja auch bei Diskettenbetrieb möglich ist?*

*2. Mich wundert übrigens, daß so schöne Befehle wie zum Beispiel MERGE (Mischen von Programmen) ebenfalls nur mit Diskettenstation verwendet werden können. Wer weiß, wie man diese Befehle auch für Kassettenrecorderbetrieb zugänglich machen kann?*

*3. Weiterhin interessiert mich, wofür die beschriftbare Schiene über der Tastatur bestimmt ist. Mit FCTN kann man einige Funktionen direkt abrufen. Ist es möglich auch selbst solche Belegungen zu definieren, zum Beispiel die Kombination CTRL 1 mit RUN zu belegen?*

*4. Schließlich möchte ich gerne wissen, ob man in einem Extended-Basic-Programm den Computer anweisen kann, zum Standardbild zurückzukehren (BYE oder QUIT als Bestandteil des Programms).*

*5. Wie kann man außerdem 40 Zeichen pro Zeile auf den Bildschirm bringen, wie in zahlreichen TI-Prospekten angegeben wird?*

*(Jens-Ulrik Buffi)*

Zu 1. und 2.: Die gewünschten Routinen existieren tatsächlich nur für den Betrieb mit Diskettenstation. Sie sind nämlich Bestandteil des ROMs im Disketten-Controller. Man könnte sie bestenfalls als Maschinensprach-Routinen nachbilden. Dazu ist aber entweder das Minimem oder der Editor/Assembler mit Diskettenstation nötig.

Zu 3.: Viele Programme von Texas Instruments benützen die obere Tastenreihe in Verbindung mit CTRL als Funktionstasten. Ihre Funktionen sind dann auf separaten Streifen aufgedruckt, die in die Schiene eingelegt werden können. Selbstverständlich kann eine ähnliche Belegung auch in eigenen Programmen erfolgen, da jede der oberen Tasten in Kombination mit FCTN oder CTRL einen speziellen Code produziert. Durch CALL KEY-Abfragen kann dieser Code geprüft und für einen entsprechenden Sprung im Programm benützt werden. Was aber nicht geht: Belegungen mit Befehlen der Art QUIT, INsert, DELete, die im Edit-Modus benützt werden können, sind nicht möglich, da diese auf der Ebene des Basic-Interpreters und damit im ROM liegen müßten. Besonders intensiv wird diese Belegung mit Funktionen aber bei Maschinencode-Programmen genutzt (zum Beispiel beim TI-Writer).

Zu 4.: Eine derartige Möglichkeit ist uns nicht bekannt, müßte aber mit entsprechenden LOAD-Werten (entspricht dem POKE anderer Computer) realisierbar sein. Vielleicht kann hier ein Leser weiterhelfen.

Zu 5.: Um 40 Zeichen pro Zeile wiedergeben zu können, muß auf Maschinensprach-Ebene der Videocontroller umgeschaltet werden. Ein Programm hierfür (Voraussetzung: Minimem-Modul) wurde in unserer Schwesterzeitschrift Computer persönlich, Ausgabe 21/83 auf Seite 142 veröffentlicht.

## Quassel-Ecke

Wer Kontakt für Datenverbindungen per Akustikkoppler und Heimcomputer sucht, oder einfach nur am Thema interessiert ist, kann uns schreiben. An dieser Stelle wollen wir Adressen und Telefonnummern von allen an dieser Technik interessierten Lesern veröffentlichen. Vergessen Sie aber bitte nicht, Ihren Computertyp und Ihren Interessenschwerpunkt anzugeben. Schreiben Sie uns auch, ob Sie bereits mit einem Akustikkoppler arbeiten. Je mehr sich melden, desto mehr Spaß macht es allen Beteiligten. Die »Quasselstrippe« blüht auf!

(lg)

Mit Interesse haben wir Ihren Artikel »Quasselstrippe« gelesen. Wir sind ein C 64-Userclub, der sich seit einiger Zeit intensiv mit der neuen Technik der DFÜ per Telefon beschäftigt. In unserem monatlichen Club-Info berichten wir ausführlich über DFÜ und die damit zusammenhängenden Probleme — natürlich meist auf den C 64 bezogen. Wir planen, unseren Mitgliedern per Telefon Software zu schicken. Ein Terminalprogramm, das den C 64 in ein intelligentes Terminal verwandelt, wurde von uns bereits erstellt.

Speziell für DFÜ-Neulinge haben wir ein DFÜ-Sonderinfo mit allen Informationen herausgegeben, die man braucht, um einen C 64 online schalten zu können. Das Info enthält auch eine Beschreibung unseres Terminalprogramms. Interessierte können es gegen 3 Mark in Briefmarken bei uns bestellen. Anfragen bitte an: Interface Computer Club, Hindenburgstr. 98, 2120 Lüneburg, oder telefonisch (04131/37876).

Ich besitze folgende Computer und Peripherie: CBM8032, CBM4032, CBM 3000, C64, Floppy SFD 1001 (ds/dd), VC1541 und Datasette, Drucker Epson FX80, Modem Epson oder Tandy (in Kürze). Meine Interessen: Business-Anwendung, Experimente und Amateurfunk.

Rolf-Thomas Clever (DL1 KAR), Mauspfad 8, 5090 Leverkusen 1, Telefon 0214/ 7 72 47

Habe Commodore 64 und Floppy 1541. Mein Interesse: Experimente. Ich besitze noch keinen Akustikkoppler, will mir aber bald einen zulegen.

Claude Diderich, 64 rue de Schoenfels, L-8151 Bridel, Telefon 00352/33 03 22

Das Thema Akustikkoppler interessiert mich. Ich besitze einen Commodore 64 mit Floppy. Ich beschäftige mich hauptsächlich mit Experimenten, Spielen und Grafik. Einen Akustikkoppler habe ich im Moment noch nicht.

Eberhard Pöthing, An der langen Hecke 109, 8861 Wechingen, Telefon 09085/477 Ich möchte mich an Ihrer Aktion »Quasselstrippe« beteiligen. Ich besitze einen Sharp MZ-731 und suche eine RS232C-Schnittstelle zum Anschluß eines Akustikkopplers. Bin an Programm- und Erfahrungsaustausch interessiert.

Günter Pflaum, Mergelshand 41, 4030 Ratingen 8, Telefon 02102/ 5 22 75

Mein Computer: Commodore 64. Meine Interessen: Business-Anwendung und Spiele. Ich besitze bis jetzt noch keinen Akustikkoppler.

Michael Schaffelhofer, Gamlingerstr.3, 8901 Aindling, Telefon 08237/12 06

Ich besitze einen C64 mit Floppy 1541 und Drucker sowie ein Tandy-Modem. Meine Interessen sind: Experimente, Hardwarebau, Anwenderprogramme in Basic und Assembler. Habe fünf Jahre Erfahrung mit Personal- und Heimcomputern.

Rudolf Möllenbeck, Alte Linnerstr. 24, 4150 Krefeld 1, Telefon 02151/2 01 30

Ich bin an Kontakten zu anderen Besitzern von Akustikkopplern interessiert. Der Computertyp: TI 99/4A. Als Akustikkoppler dient ein Tandy AC-3. Ich interessiere mich für Experimente und Business-Anwendungen, sowie allgemeinen Erfahrungsaustausch.

Werner Völker, Römerstr. 43, 7336 Uhingen, Telefon 07161/3 15 21

Bin an Datenfernübertragung interessiert und habe gelesen, daß ein Telefon-Modem für den PC-1500 herausgebracht werden soll. Wer weiß Bescheid?
Karl-Heinz Krönes, Nußdorfstr. 6, 7251 Weissach, Telefon 3 18 67

Ich besitze einen 48-KByte-Spectrum und bin am Einsatz von Akustikkopplern interessiert. Im Augenblick habe ich noch keinen. Meine Interessenschwerpunkte sind: Basic, Tabellenkalkulation, Balken-Kreisdiagramme, Adressen und Datenverwaltung, Business-Anwendungen.

Michael Zinner, Bettinastr. 31, 6000 Frankfurt/M. 1, Telefon 0611/75 12 55

## Schnelle Hardcopy

(Ausgabe 3/84, Seite 64ff)
Im Programm können je nach verwendeter Speichererweiterung Fehler auftreten. Bei einer Speichererweiterung von 8 oder mehr KByte ist alles wie beschrieben in Ordnung.

In der Grundversion des VC20 ist bei Beachtung der Hinweise in Zeile 2 des Programms ebenfalls keine Schwierigkeit zu erwarten. Anders sieht die Sache aber aus, wenn man das Hardcopy-Programm als Subroutine mit MERGE (POKE43,1: POKE44,16) an ein bestehendes Programm anhängt. In diesem Fall muß zunächst das geschützte Merge-Programm in den Arbeitsspeicher geholt werden, bevor das Hardcopy-Programm gestartet wird.
Verwendet man das Grafik-Modul VC-1211 (allein!), so ist anders als in der Beschreibung folgendes zu beachten: Vor dem Einladen muß man mit »GRAPHIC1« den Grafikmodus einschalten und mit »GRAPHIC0« (nicht mit »GRAPHIC4«!) in den Textmodus zurückkehren. Ist dann das Programm geladen, so ergibt »PEEK(56)« den Wert 16 und »PEEK(648)« den Wert 30. Damit würde aber das unveränderte Programm in eine Schleife geraten (Zeile 4 und 8). Deshalb muß die Zeile 8 neu lauten:
8 IFPEEK(56)(=16THEN6
Verwendet man das Modul VC-1211A allein, dann sollte man alle nicht unbedingt notwendigen Zeilen entfernen, nachdem man das Programm geprüft hat, ob es richtig läuft. So spart man viel Platz für aufwendige Grafikprogramme. Es lassen sich ohne Probleme folgende Zeilen entfernen: 0 bis 3, 30, 50, 60 und 70.

(Dr. C. Roosen-Runge)

## Mit Charme + Chip

(Ausgabe 6/84, Seite 97)
Auf Seite 98 muß im Text rechts unten die Zeile 9755 statt
»9755 PRINT AT 3,2;« Sie haben;LEN TS$-1;:IF LEN T$2 OR LEN T$=1 THEN PRINT »Töne«: GOTO«
»9755 PRINT AT 3,2;« Sie haben;LEN T$-1;:IF LEN T$>2 OR LEN T$=1 THEN PRINT »Töne;« GOTO 9757« heißen.

## Gruseleien im Monster-Labyrinth

(Ausgabe 5/84, Seite 56ff)
Damit man mit der Figur nicht nur links und rechts aus dem Bild fahren kann, sondern auch nach oben und unten (ohne gleich die Meldung »Out of Screen» zu erhalten) muß man die Zeilen 410, 411, 620 und 720 ergänzen indem man folgende Änderungen am Listing durchführt:
410 IF h)30 THEN LET h=0 : IF v)20 THEN LET v=0
411 IF h(0 THEN LET h=31 : IF v(0 THEN LET v=21
620 LET v=v+1 : IF v=22 THEN LET v=0
620 LET v=v-1 : IF v=0 THEN LET v=21

(Reinhard Wacker)

## PRINT USING für Commodore 64

(Ausgabe 3/84, Seite 52/53)
Aller Vorsicht zum Trotz habe ich dem Commodore 64 die Möglichkeit gelassen, in Extremfällen seinen »widerlichen« Abbruchfehler auszuspielen, und zwar durch den Einsatz von 10↑UR in Zeile 60030. Ich bin »in mein eigenes Messer gelaufen«, wie ein Anrufer meinte. Vielen Dank für den Anruf! Hier die Korrektur:
60026 IF UR=0 THEN UR=1 : GOTO 60030
60027 UI=UR: UR=1: FOR UI=
UI TO 1 STEP-1: UR=UR*10: NEXT UI

Dafür muß man zweimal in Zeile 60030 den Faktor: »10↑« löschen

Die komplizierte Konstruktion mit Rückwärtsschleife wurde gewählt, um eine zusätzliche Hilfsvariable zu vermeiden. Außerdem hat sich in der linken Textspalte, Seite 53. ein Druckfehler eingeschlichen: Die Formel für zweistellige Rundung heißt nur »INT(100*A+.5)/100«, also bitte »*A)« am rechten Ende streichen.

(Prof. W. Tosberg)

## Listing des Monats: Fallobst

(Ausgabe 6/84, Seite 14)
Bei diesem Listing fielen mir einige Schönheitsfehler auf. Statt nach zwei Tagen und Nächten werden die Fehler erst nach der doppelten Zeit gelöscht. Verbesserung: »1470 GOTO 1510« statt »1470 GOTO 1560«.

Nach der Frage »Nochmals j n?« und der Antwort »J» soll das Spiel von vorne beginnen. Es werden jedoch nur die Zahl der Körbe auf fünf und der Score auf Null gesetzt. Es beginnt auch kein neuer Tag. Um dies zu ändern, muß man folgende Zeilen einfügen:
2465 CALL HCHAR(2,MS, MG)
2472 TZAEHL=0
2473 F1=0
2474 GOSUB 2050
2475 MOND=147
2476 MG=130

(Michael Ruck)

## Hilfe für TI-Benutzer

Nachtrag zur Ausgabe 5/84 (Marktübersicht zu neuer TI 99/4A-Software: Businessprogramme wie zum Beispiel Lagerverwaltung bietet Digisoft an.

Info: Digisoft, Krampnitzer Weg 17a, 1000 Berlin 22, Tel. (030) 3652347 (10.00 bis 22.00 Uhr)

## pool-billard

(Ausgabe 5/84, Seite 58ff)
Durch einige Änderungen kann das Programm sehr vereinfacht und gekürzt werden. Die Eingabe der Teilnehmerzahl und der Namen ist viel zu umständlich (Zeile 350 bis 530). Durch den ON...GOTO-Befehl und einfache mathematische Konditionierungen läßt sich diese Routine um mehr als die Hälfte kürzen.

Nach Abfrage der Spielerzahl folgt vor der CALL-KEY-Routine eine Warteschleife (Zeile 280 bis 350). Wozu?

Die IF...THEN-Konditionierungen in den Zeilen 530 bis 720 sind ebenfalls unrationell. Das ELSE fällt durch eine einfache Umkehrung der Konditionierungen weg.
Beispiel: »540 IF B=4 THEN 550 ELSE 590« wird zu »540 IF B≠4THEN 590«.

Das Spielfeld wird bei jedem neuen Stoß neu hingemalt, nachdem der Name des Spielers kurz vorher angezeigt worden ist (Zeile 530 bis 1060). Dies läßt sich auch durch Löschen des Namens und des letzten Loches allein umgehen. Der Name muß dann mittels HCHAR in Zeile 24 hingemalt und auch wieder gelöscht werden. So wird auch das Scrollen des Bildschirms verhindert. Durch Einfügen von »725 IF KENN≠0 THEN1070« und »1065 KENN=1« wird dann auch noch das langweilige und zeitraubende Löschen des Bildschirms vermieden.

In den Zeilen 800 bis 1060 werden die Zeilennummern für das Spielfeld mit Hilfe von VCHAR hingemalt. Dies ist durch ein DATA-Feld wesentlich speicherplatzfreundlicher zu realisieren.

Die Zeile »1080 C=INT (22—13+1)*RND)+13« läßt sich auch rationeller schreiben: »1080 C=INT(10*RND) +13«. Ebenso verhält es sich mit Zeile 1200.

Die IF...THEN-Abfragen in den Zeilen 1490 bis 1520 lassen sich durch ON...GOTO auf eine Zeile reduzieren. Die Sound-Routinen für Treffer und Fehlstöße (Zeile 1530 bis 2560) lassen sich durch Verschieben ans Ende des Programms und durch Aufruf mit GOSUB kürzen.

Die Eingabe der Strings für die Spielernamen (A1$, etc.) lassen sich wie auch die Treffer- und Fehlstoßvariablen durch Feldvariablen rationeller gestalten, zum Beispiel A$(1), T(1) und F(1). Dadurch ist die Abfrage der Spielernamen in einer FOR... NEXT-Schleife möglich, die Ausgabe der Spielernamen und der Ergebnisse ebenfalls. Außerdem wird das Nullsetzen der Variablen für ein neues Spiel (Zeile 2890 bis 3090) wesentlich kürzer.

Es sind aber noch viele weitere Vereinfachungen möglich.

Der Autor kann wohl kaum behaupten, er habe das Programm so gestalten wollen, daß es für jeden verständlich und nachvollziehbar sei, denn verständlich und nachvollziehbar ist es durch seine Umständlichkeit lediglich für Computerfans mit längerer Erfahrung, nicht jedoch für Anfänger, die etwas lernen wollen.

TI-Besitzer, die an der verbesserten »pool-billard«-Version interessiert sind, möchten sich bitte an folgende Adresse wenden: Alfred Manthey Rojas, Eichenkamp 22, 2200 Elmshorn, Tel. (04121) 74734

(Alfred Manthey Rojas)

# Russisch aus dem Computer

**Russisch-Kurse beginnen mit den kyrillischen Buchstaben. Eine Lektion, an der schon viele scheiterten. Hier hilft ein Lernprogramm für den TI 99/4A weiter.**

Grundlage dieses Programms ist die Zuordnung der lautlichen Entsprechungen zu den russischen Schriftzeichen. Dazu werden im ersten Teil des Programmes (-zdef-; Zeile 40 bis 100) in zusammenhängender Reihenfolge die russischen Zeichen von Code 65 bis 96 definiert. Diese Folge entspricht den Großbuchstaben auf der Tastatur und einigen Sonderzeichen. Weiterhin werden in diesem Programmteil einige Sonderzeichen definiert, die später für die Lauttabelle benötigt werden.

Diese Lauttabelle wird im 2. Programmteil (-ltab-; Zeile 105 bis 108) erstellt und besteht aus zwei Feldern. Das erste Feld (UGS$(I)) enthält in der Reihenfolge der Zeichencodes der russischen Schriftzeichen deren (ungefähre) lautliche Entsprechung in Normalschrift (TI-Kleinbuchstaben). Das zweite Feld enthält die entsprechenden Zeichen in vereinfachten phonetischen Symbolen, die speziell für diejenigen gedacht sind, die die Lautschrift einigermaßen beherrschen.

Zur Erstellung dieser Lauttabelle wird ein etwas ungewöhnlicher Weg gewählt: Da die Codes der phonetischen Symbole meistens größer als 127 sind und daher im Listing nicht mehr eindeutig erkennbar sind, werden die Symbole nicht als direkte Stringzeichen abgelegt. Stattdessen wird der Code für jedes Zeichen auf eine Länge von 3 Stellen festgelegt und in den DATA-Zeilen abgelegt und später mit CHR$(VAL...) eingelesen (Zeile 140 bis 180). Dieser Vorgang ist zwar speicherplatz- und zeitaufwendiger, aber auch gleichzeitig wesentlich eingabefreundlicher. Die einzige Ausnahme stellt Zeile 1220 dar. Alle dort unterstrichenen Zeichen müssen über die »CTRL«-Taste eingetippt werden.

Nach diesem längeren Vorspann beginnt das eigentliche Programm. Im Hauptmenü (-hmen-; Zeile 185 bis 240) kann durch Tastendruck zwischen »Einzelzeichen«, »Vokabeltraining«, »Vorsprechen«, »Ende« und »Tastenbelegung« gewählt werden.

Wahl »5« bringt die Belegung der Tastatur mit den russischen und phonetischen Zeichen auf den Bildschirm und dient zur Orientierung bei der Eingabe. Dieses Unterprogramm (-tbel-; Zeile 1165 bis 1250) kann auch später immer wieder aufgerufen werden, wenn eine russische oder phonetische Eingabe verlangt wird, indem man einfach »help« (in Kleinbuchstaben!!) eingibt.

Wahl »4« (-end-; Zeile 740 bis 755) veranlaßt den Computer, sich höflich auf Russisch zu verabschieden und das Programm zu beenden.

Wahl »1« bis »3« sind die eigentlichen Lern- und Trainingsteile.

Wahl »1« (-eint-; zeile 245 bis 375) erzeugt eine wählbar lange Schleife, in der zufällig ausgewählte russische Einzelzeichen angezeigt werden. Dabei muß deren lautliche Entsprechung entweder in Normalschrift (Kleinbuchstaben) oder in Lautschrift (mit vorangestelltem #) eingegeben werden. Je nachdem, ob die Eingabe mit der internen Lauttabelle übereinstimmt, wird entweder das nächste Zeichen abgefragt oder zuerst die richtige Lösung angezeigt und der Fehlerzähler erhöht. Nach dem Beenden der Schleife erfolgt die Anzeige der gemachten Fehler (-faus-; Zeile 800 bis 830) und der Rücksprung ins Hauptmenü.

Wahl »2« (-vokt-; Zeile 380 bis 660) bringt zunächst ein Submenü, in dem eine Kombination von Aussprache- und Vokabel-

| Code | Aufschrift | Zeichen |
|---|---|---|
| 65 | A | А |
| 66 | B | В |
| 67 | C | С |
| 68 | D | А |
| 69 | E | Е |
| 70 | F | Г |
| 71 | G | Ь |
| 72 | H | Н |
| 73 | I | Ы |
| 74 | J | ё |
| 75 | K | К |
| 76 | L | Л |
| 77 | M | М |
| 78 | N | И |
| 79 | O | О |
| 80 | P | Р |
| 81 | Q | Ф |
| 82 | R | Я |
| 83 | S | Э |
| 84 | T | Т |
| 85 | U | Ц |
| 86 | V | У |
| 87 | W | Ш |
| 88 | X | Х |
| 89 | Y | Ү |
| 90 | Z | Ж |
| 91 | [ | Б |
| 92 | \ | Ю |
| 93 | ] | З |
| 94 | ^ | П |
| 95 | - | Щ |
| 96 | ` | Й |

**Tastatur-Belegung**

Der Belegung der Tasten mit den russischen Zeichen liegen folgende Kriterien zugrunde:

1) Tastenschrift entspricht dem russischen Zeichen:
A, B, C, E, H, K, M, O, P, T, X, Y

2) Tastenschrift ähnelt dem russischen Zeichen:
►Г И О Я Ц Ш Б П

3) Tastenschrift entspricht dem Lautwert:
►А Л Ы З

4) Willkürliche Zuordnung:
►Ь ё у Ж Ю Щ Й

Als Faustregel gilt:
Russische Zeichen = Alle TI-Großbuchstaben und sechs Sonderzeichen ([, ], ^, -, `)
Phonetische Zeichen = als TI-Kleinbuchstaben: v,

**Hinweise für die Tastatur-Belegung**

s, j, x, z, u, p, i
über CTRL+ entsprechende Taste: a, b, d, e, f, g, k, l, m, n, r, t
willkürlich: Ё X З О

**Variablenliste**

| | |
|---|---|
| I,J,II: | Allgem. Laufvariablen |
| T,S,K: | Tastaturabfrage |
| DEF$: | Definitionscode |
| N: | Zeichencode |
| A$,B$: | Stringcodes |
| L: | Durchlaufwahl |
| ZC: | Zufallszeichencode |
| LOES$: | Lösungseingabe |
| FZ: | Fehlerzähler |
| XVOK: | Vokabelauswahl |
| RX$: | Russ. Zeichenkette |
| UEBERS$: | Übersetzung |
| LU$,LP$: | Aussprache |
| LUEB$: | Übersetzungseingabe |
| DEL: | Verzögerung |
| X$: | Einzelzeichenzwischenspeicher |
| X1: | ASCII-Wert von X$ |
| X: | Tabellenzeiger |

**Programmbeschreibung**

| | |
|---|---|
| 10-35: | Deklarationen |
| 40-100: | Zeichendefinition (-zdef-) |
| 105-180: | Lauttabelle erstellen (-ltab-) |
| 185-240: | Hauptmenü (-hmen-) |
| 245-375: | Einzelzeichen (-eint-) |
| 380-660: | Vokabeln (-vokt-) |
| 380-435: | Submenü |
| 440-470: | Vokabelwahl |
| 660: | Vokabeldaten |
| 665-735: | Vorsagen (-vors-) |
| 740-775: | Programmende (-end-) |
| 780-795: | Verzögerung (-verz-) |
| 800-830: | Fehleranzeige (-faus-) |
| 835-1160: | Lautfolge (-phon-) |
| 1165-1250: | Tastenbelegung (-tbel-) |

training und die Anzahl der Durchläufe gewählt werden kann. Danach wird aus einer internen DATA-Liste ein russisches Wort und dessen Übersetzung per Zufallsgenerator ausgewählt. Zu dem russischen Wort werden in einem Unterprogramm (-phon-; Zeile 835 bis 1160) zwei Strings erzeugt, die der Aussprache in Normal- und Lautschrift entsprechen. Danach wird das russische Wort angezeigt und je nach Unterprogramm-Wahl die Eingabe der Aussprache und/oder Übersetzung verlangt, die dann wie in Wahl »1« ausgewertet wird. Zusätzlich werden dann zur Vertiefung des Lerneffekts Wort, Übersetzung und Aussprache in einer Art Vokabelkarte nochmals angezeigt. Nach Beenden der Schleife erfolgen Fehleranzeige und Rücksprung wie bei Option »1«.

Mit Wahl »3« können beliebige russische Zeichenfolgen eingegeben werden, die dann im Unterprogramm -phon- in ihre lautlichen Entsprechungen umgewandelt und angezeigt werden .

Das Herz der beiden letzten Programmteile ist das Unterprogramm -phon-, welches die Lautfolge erstellt. Dabei wird jedes russische Zeichen einzeln ausgewertet. Wenn es für das jeweilige Zeichen nur eine Realisierung gibt, werden die Entsprechungen aus der Lauttabelle an Aussprachestrings (LU$,LP$) angehängt (Zeile 910/915). Bei den anderen Zeichen werden die umgebenden Buchstaben berücksichtigt und die sich daraus ergebenden Kombinationen individuell erstellt (Zeile 925 bis 1145).

Erfaßt werden von diesem Programm:
— Ь entfällt nach ★, Ш, У, Щ, (Zeile 925)
— Л vor Я, Е, И, е, Й, Ю, als »weiches l«, sonst als »hartes l« (Zeile 1025 bis 1055), dieser Unterschied ist nur in der Lautschrift erkennbar.
— В am Wortende und vor stimmlosen Konsonanten als »ff«, sonst als »w« (Zeile 1060 bis 1095).
— Ч im Wort УТО als »sch« (Zeile 1125 bis 1135)
— Г vor К als »ch«, in den Silben ЕГО und ОГО als »w«, sonst als »g« (Zeile 950 bis 1020)
— ungültige Zeichen werden ignoriert (Zeile 905), ein Leerfeld wird jedoch als Leerfeld wiedergegeben (Zeile 935 bis 940)
— nicht unterschieden werden allerdings Vokale in betonten und unbetonten Silben! Bei О werden die beiden Möglichkeiten als »a/o« angegeben.

Anmerkungen zur Lauttabelle:
1) Das Vokabelsystem bezüglich der Längenzeichen und der i-Laute wird etwas vereinfacht.
2) Erweichte Konsonanten (das heißt mit nachgestelltem Ь) werden nicht durch ihre Sonderzeichen (zum Beispiel k, t) wiedergegeben. Stattdessen wird das Ь durch ein »j« dargestellt, was auch der sprachlichen Realisierung am nächsten kommt.
3) Die Unterscheidung zwischen stimmhaften und stimmlosen Konsonanten ist nur in der Lautschrift zu erkennen.

Bemerkungen zur Vokabelliste:

Die Vokabelliste ab Zeile 660 ist zur selbständigen Erweiterung durch den Anwender gedacht. Zur Erweiterung ist es notwendig, die russischen Schriftzeichen zunächst in den TI-Zeichensatz umzuwandeln (hierbei ist Tabelle 1 sehr nützlich!). Diese Zeichenfolge muß dann in Großbuchstaben in die entsprechende DATA-Zeile 660 eingetragen werden, gefolgt von der deutschen Übersetzung (in Kleinbuchstaben!). Danach muß in Zeile 455 noch die entsprechende Anzahl der Wortpaare eingesetzt werden. Im Listing sind zu Testzwecken die Worte DA (ja) und HET (nein) vorgegeben, entsprechend lautet Zeile 455 »XVOK=INT(RND*2)+1«. Wenn jetzt zum Beispiel diese Liste um das Wort ГАЗЕТА erweitert werden soll, so ist in die DATA-Zeile 660 die Folge »..,FASETA,zeitung« anzuhängen und in 455 statt der 2 eine 3 zu schreiben.

Für die Vokabeln stehen noch mehr als 4 KByte RAM zur Verfügung, den Platz für die DATA-Zeilen schafft man sich mit »RESEQUENCE«. Zu beachten ist, daß keine DATA-Zeile vor die Listingzeile 660 gestellt wird, andernfalls muß auch die Zeilennummer in 450 geändert werden!

Schlußbemerkung zum Lernen mit dem Computer:

Prinzipiell sei gesagt: Das Programm »Russisch Trainer« kann und soll keinen Russischkurs ersetzen! Einmal, weil es wichtige Aspekte der Sprache, wie die Grammatik, vernach-

lässigt und weil zum anderen eine Sprache erst dadurch lebendig wird, daß sie gesprochen wird (und zwar im wahrsten Sinne des Wortes)!

Wer aber Russisch lernen will, sei es im Selbststudium mit Schallplattenkurs oder auf der Volkshochschule, dem hilft dieses Programm besonders in der Anfangsphase beim Erlernen der Schriftzeichen oder später beim Vokabeln lernen.

Wer aber nach dem Motto »Programm laden und dann kann ich Russisch« handelt, dem kann nicht geholfen werden!

(Michael Anton/wb)

```
10 REM * RUSSISCH-TRAINER *
15 REM * by michael anton *
20 CALL CLEAR
25 PRINT TAB(7);"bitte warten": : : : :
: : : : :
30 CALL SOUND(-4250,440,9)
35 DIM UGS$(32),PHON$(32)
40 REM -zdef-
45 DATA 68,001C1414247E4242,70,007C42404
040404,71,0040404078444478,73,0042424272
4A4A72,74,440038444478403C
50 DATA 76,000E0A0A12122262,78,004143454
9516141,81,00083E49493E0808,82,001C24241
C0C1424,83,0038440418044438
55 DATA 85,0044444444447F01,86,002222261
A020202,87,0049494949497F,90,00086B2A1
C2A4949,91,007C404078444478
60 DATA 92,00404E5161514E40,93,003844041
C044438,94,007C444444444444,95,005454545
4547F01,96,140842464A526242
65 DATA 129,0000000034C4C34,130,0040404
078444478,131,0000001820182018,132,00040
404344C4C34,133,000038444478403C
70 DATA 134,001820207020206,135,00304848
3808483,136,0000000008142241,139,0040404
850605048,140,0020202020202018
75 DATA 141,000000005A4A4A4A,142,0000000
050684848,145,0000003844046438,146,00000
06C3020202,148,0020207020202018
80 DATA 156,003048081008502,158,000C0A08
08082818,137,0010143850101000
85 FOR I=1 TO 38
90 READ N,DEF$
95 CALL CHAR(N,DEF$)
100 NEXT I
105 REM -ltab-
110 DATA 097,129,119047102102,118047134,
115115,115,100,132,106101,106133,1030470
99104047119
115 DATA 135047120047118,106,106,110,142
,105,105,106111,106111,107,139,108,140,1
09,141,105101,105
120 DATA 111047097,145047136,114,146,102
,134,106097,106129,115,122,116,148,122,1
48115
125 DATA 116115099104,148158,115099104,1
58,099104,120,117,117,115099104,156,098,
130,106117,106117
130 DATA 101,131,112,112,115099104116115
099104,158148158,105,105
135 CALL SOUND(-4250,880,6)
140 FOR I=1 TO 32
145 READ A$,B$
150 FOR J=1 TO LEN(A$)STEP 3
155 UGS$(I)=UGS$(I)&CHR$(VAL(SEG$(A$,J,3
)))
160 NEXT J
165 FOR J=1 TO LEN(B$)STEP 3
170 PHON$(I)=PHON$(I)&CHR$(VAL(SEG$(B$,J
,3)))
175 NEXT J
180 NEXT I
185 REM -hmen-
190 CALL CLEAR
195 CALL SCREEN(16)
200 PRINT TAB(3);"russisch-trainer": : :
:"1 -> einzelzeichen": :"2 -> vokabeln"
: :"3 -> vorsprechen": :
205 PRINT "4 -> ende": :"5 -> tastenbele
gung": : :
210 CALL KEY(0,T,S)
215 CALL SOUND(-10,440,9)
220 IF S=0 THEN 210
225 CALL SOUND(1000,220,6,440,7,880,8)
230 IF (T<49)+(T>53)THEN 210
235 ON (T-48)GOSUB 245,380,665,740,1165
240 GOTO 185
245 REM -eint-
250 CALL CLEAR
255 CALL SCREEN(3)
260 INPUT "wieviele buchstaben? ":L
265 FZ=0
270 FOR I=1 TO L
275 RANDOMIZE
280 ZC=INT((RND*32)+1)
285 CALL CLEAR
290 PRINT "die lautliche entsprechung":
:"von",CHR$(ZC+64): :"ist:": : : :
295 PRINT "bei phonet. eingabe bitte # v
oranstellen.": :
300 INPUT LOES$
305 IF LOES$<>"help" THEN 320
310 GOSUB 1165
315 GOTO 285
320 IF LOES$=UGS$(ZC)THEN 330
325 IF POS(LOES$,PHON$(ZC),2)=0 THEN 345
330 PRINT "richtig!!!!!!": : : : :
335 CALL SOUND(1000,880,0)
340 GOTO 365
345 PRINT "falsch!!!!!!!": : :
350 CALL SOUND(500,220,0)
355 PRINT "richtig waere gewesen:": :TAB
(7);UGS$(ZC): :TAB(5);"oder": :TAB(7);"#
";PHON$(ZC): :
360 FZ=FZ+1
365 GOSUB 780
370 NEXT I
375 GOTO 800
380 REM -vokt-
385 CALL CLEAR
390 CALL SCREEN(10)
395 PRINT "1 -> aussprache": :"2 -> auss
prache + vokabeln": :"3 -> vokabeln": :
: :
400 CALL KEY(0,T,S)
405 CALL SOUND(-10,440,9)
410 IF S=0 THEN 400
415 CALL SOUND(1000,220,6,440,7,880,8)
420 IF (T<49)+(T>51)THEN 400
425 CALL CLEAR
430 INPUT "wieviele woerter sollen ge- p
rueft werden? ":L
435 FZ=0
440 FOR II=1 TO L
445 RANDOMIZE
450 RESTORE 660
455 XVOK=INT(RND*2)+1
460 FOR J=1 TO XVOK
465 READ RX$,UEBERS$
470 NEXT J
475 GOSUB 835
480 ON (T-48)GOSUB 510,590,595
485 CALL CLEAR
490 PRINT RX$,UEBERS$: : :LU$,LP$: : : :
:
495 GOSUB 780
500 NEXT II
505 GOTO 800
510 CALL CLEAR
515 PRINT "wie wird": :RX$: :"ausgesproc
hen?": : :"(bei phonet. eingabe bitte #v
oranstellen)": :
520 INPUT LOES$
525 IF LOES$<>"help" THEN 540
530 GOSUB 1165
```

```
535 GOTO 510
540 IF LOES$=LU$ THEN 550
545 IF POS(LOES$,LP$,2)=0 THEN 565
550 PRINT : : :"richtig!!": :
555 CALL SOUND(1000,880,0)
560 GOTO 580
565 PRINT : :"falsch!!": :
570 CALL SOUND(500,220,0)
575 FZ=FZ+1
580 GOSUB 780
585 RETURN
590 GOSUB 510
595 CALL CLEAR
600 PRINT "was heisst": :RX$: :"auf deut
sch?": : : : : :
605 INPUT LUEB$
610 IF LUEB$<>UEBERS$ THEN 630
615 PRINT : :"richtig!!": :
620 CALL SOUND(1000,880,0)
625 GOTO 645
630 PRINT : :"falsch!!": :
635 CALL SOUND(500,220,0)
640 FZ=FZ+1
645 GOSUB 780
650 RETURN
655 REM -vokdat-
660 DATA DA,ja,HET,nein
665 REM  -vors-
670 CALL CLEAR
675 CALL SCREEN(12)
680 PRINT "welches wort soll umgewan-":
:
685 INPUT "delt werden? ":RX$
690 IF RX$<>"help" THEN 705
695 GOSUB 1165
700 GOTO 670
705 GOSUB 835
710 PRINT : : :"die aussprache ist:": :L
U$:"in normaler schreibweise und": :LP$:
"in lautschrift.": : :
715 PRINT "neues wort: {space}": :
720 CALL KEY(0,T,S)
725 IF S=0 THEN 720
730 IF T=32 THEN 665
735 RETURN
740 REM  -end-
745 CALL CLEAR
750 CALL SCREEN(14)
755 PRINT TAB(7);"DO CBNDAHNR!!!": : : :
 : : : : : : :
760 CALL SOUND(1000,880,2)
765 GOSUB 780
770 CALL CLEAR
775 END
780 REM -verz-
785 FOR DEL=1 TO 1000
790 NEXT DEL
795 RETURN
800 REM -faus-
805 CALL CLEAR
810 CALL SCREEN(6)
815 CALL SOUND(1000,220,6,440,7,880,8)
820 PRINT "bei ";L;" aufgaben haben sie"
: :"insgesamt ";FZ;" fehler.": : : : : :
 : :
825 GOSUB 780
830 RETURN
835 REM  -phon-
840 LP$=""
845 LU$=""
850 FOR I=1 TO LEN(RX$)
855 X$=SEG$(RX$,I,1)
860 X1=ASC(X$)
865 X=X1-64
870 IF X1=66 THEN 1060
875 IF X1=32 THEN 935
880 IF X1=70 THEN 950
885 IF X1=71 THEN 925
890 IF X1=76 THEN 1025
895 IF X1=79 THEN 1100
900 IF X1=86 THEN 1125
905 IF (X1<65)+(X1>96)THEN 1155
910 LP$=LP$&PHON$(X)
915 LU$=LU$&UGS$(X)
920 GOTO 1155
925 IF POS("UVW_",SEG$(RX$,I-1,1),1)=0 T
HEN 910
930 GOTO 1155
935 L1$=" "
940 L2$=" "
945 GOTO 1140
950 IF I=LEN(RX$)THEN 975
955 IF ASC(SEG$(RX$,I+1,1))<>75 THEN 975
960 L2$="x"
965 L1$="ch"
970 GOTO 1140
975 IF I+1<>LEN(RX$)THEN 995
980 IF ASC(SEG$(RX$,I+1,1))<>79 THEN 995
985 IF ASC(SEG$(RX$,I-1,1))=69 THEN 1010
990 IF ASC(SEG$(RX$,I-1,1))=79 THEN 1010
995 L2$=CHR$(135)
1000 L1$="g"
1005 GOTO 1140
1010 L2$="y"
1015 L1$="w"
1020 GOTO 1140
1025 IF POS("EJNR\'",SEG$(RX$,I+1,1),1)=
0 THEN 1045
1030 L2$=CHR$(140)
1035 L1$="l"
1040 GOTO 1140
1045 L2$=CHR$(137)
1050 L1$="l"
1055 GOTO 1140
1060 IF POS("^QKTWUV_",SEG$(RX$,I+1,1),1
)<>0 THEN 1070
1065 IF I=LEN(RX$)THEN 1070 ELSE 1085
1070 L1$="ff"
1075 L2$=CHR$(134)
1080 GOTO 1140
1085 L1$="w"
1090 L2$="v"
1095 GOTO 1140
1100 IF I=LEN(RX$)THEN 1110
1105 IF SEG$(RX$,I+1,1)=" " THEN 1110 EL
SE 910
1110 L1$="o"
1115 L2$=CHR$(145)
1120 GOTO 1140
1125 IF RX$<>"VTO" THEN 910
1130 L1$="sch"
1135 L2$=CHR$(158)
1140 LP$=LP$&L2$
1145 LU$=LU$&L1$
1155 NEXT I
1160 RETURN
1165 REM  -tbel-
1170 CALL CLEAR
1175 PRINT "lage der russischen zeichen
1) ueber shift:": :
1180 PRINT "*!*!*!*!*!^!*!*!*!*!*":" Q!W
!E!R!T!Y!U!I!O!P!*":"  A!S!D!F!G!H!J!K!L
!*!*":" *!Z!X!C!V!B!N!M!*!*!*": :
1185 PRINT :"2) ueber fctn:": :
1190 PRINT "*!*!*!*!*!*!*!*!*!*!*":" *!*
!*![!]!*!_!*!*!*!*":"  *!*!*!*!*!*!*!*!*
!*!*":" *!\!*!`!*!*!*!*!*!*!*": :
1195 PRINT TAB(20);">>"
1200 CALL KEY(0,K,S)
1205 IF S=0 THEN 1200
1210 CALL CLEAR
1215 PRINT "lage der phonetischen sym-
bole, 1) ueber ctrl:": :
1220 PRINT "*!*!*!*!*!*!*!*!*!*!*!*":" Q!*
!E!R!T!*!*!I!*!*!*!*":"  A!*!D!F!G!H!*!K!L
!;!*":" *!*!*!C!*!B!N!M!*!*!*": :
1225 PRINT :"2) als kleinbuchstaben:": :
1230 PRINT "*!*!*!*!*!*!*!*!*!*!*!*":" *!w
!*!*!*!*!u!i!o!p!*":"  *!s!*!*!*!*!j!*!*
!*!*":" *!z!x!*!v!*!*!*!*!*!*!*": :
1235 PRINT TAB(20);">>"
1240 CALL KEY(0,K,S)
1245 IF S=0 THEN 1240
1250 RETURN
```

**Listing zum Russisch-Trainer**

# Screen Design

**Programme für die automatische Zeichendefinition beim TI 99 gibt es viele. »Screen Design« kann jedoch viel mehr. Benötigt wird ein TI 99/4A, das Extended-Basic-Modul und ein Diskettenlaufwerk.**

Das endlose Eintippen von Char-Strings gehört der Vergangenheit an. Zeichendefinition und Bildaufbau Ihrer Programme müssen Sie nicht mehr selbst eingeben, sondern können diese zeitaufwendige Arbeit dem Computer überlassen. Er berechnet aus den von Ihnen kreierten Bildern eine Merge-Datei, die Sie problemlos in Ihr Programm einbinden können.

Die Bedienung von »Screen Design« ist äußerst einfach, da alle Optionen per Menü ausgewählt werden. Nach der Wahl der Bildschirmfarbe und der Angabe, ob Sie in den oberen oder den unteren zwei Dritteln des Bildschirms zeichnen wollen, erscheint das Hauptmenü. Es hat folgende Optionen:

1 = ASCII — Der Cursor zum Zeichnen wird in das eingegebene Zeichen umgewandelt.

2 = CHAR — Mittels der Pfeiltasten (E,S,D,X) wird ein Rahmen auf das gewünschte Zeichen gesteuert und durch Drücken der Taste »ENTER« übernommen.

3 = CHAR Zeichnen — Auf einer 8 x 8-Matrix können Sie ein neues Zeichen definieren. Mit den Pfeiltasten (E,S,D,X) wird der Cursor bewegt, mit einer beliebigen Taste setzen Sie einen Punkt, mit der Taste »0« wird ein Punkt gelöscht und mit der Taste »1« beenden Sie die Definition.

4 = FARBE — Der Befehl »CALL COLOR« erscheint auf dem Bildschirm. Durch die Eingabe des Zeichensatzes und der Vorder- und Hintergrundfarbe ändern Sie die Farben der von Ihnen gewählten Zeichen.

5 = BILD wechseln — Sie können wählen, ob Sie im oberen oder unteren Bildschirmbereich zeichnen wollen.

6 = BILD sehen — Das Hauptmenü verschwindet und Sie können das gesamte Bild betrachten. Auf Tastendruck gelangen Sie wieder zum Hauptbild.

7 = BILD ablegen — Das Bild wird unter dem Namen »SCREEN« auf Diskette als Merge-Programm abgelegt.

8 = HELP — Die Helpzeile hält Sie ständig in bezug auf Cursorzeichencode, Vorder- und Hintergrundfarbe auf dem laufenden. Wenn Sie die Taste »8« drücken, können Sie mit den Pfeiltasten (E,X) in den Zeichencodes blättern. Bei Betätigung der Taste »ENTER« wird der Cursor durch das momentan angezeigte Zeichen ersetzt.

(Jürgen Wrede/Wolfgang Czerny)

**Listing für schnelle Zeichendefinition »Screen-Design«**

```
300 !---- SCREEN-DESIGN ---+
310 !                       I
320 ! WITH BASIC-GENERATOR  I
330 !      JUERGEN WREDE    I
340 !   PRODUCED MAI 1984   I
350 !-----------------------+
360 DIM B(24,28),A(8,8),CV(1
4),CH(14),G$(6),ZEI$(143)
370 CALL CHAR(42,"",43,RPT$(
"F",16),47,"FF81818181818181FF
")
380 NEU$="+*/" :: CALL CHARP
AT(43,ZEI$(43),42,ZEI$(42),4
7,ZEI$(47))
390 SPRUNG=0 :: CHAR=96 :: R
R=33 :: HEX$="0123456789ABCD
EF"
400 ON WARNING NEXT :: CALL
TITEL
410 !----colorfeld--------
420 CALL COLOR(2,2,16):: FOR
 I=1 TO 14 :: CV(I)=2 :: CH(
I)=1 :: NEXT I :: CH(2)=16
430 !----bildfeld--------
440 FOR Z=1 TO 24 :: FOR S=1
 TO 28 :: B(Z,S)=32 :: NEXT
S :: NEXT Z
450 !
460 DATA =ASKII UEBERNEHMEN,
=CHAR  UEBERNEHMEN,=CHAR  ZE
ICHNEN,=FARBE SETZEN,=BILD
WECHSELN,=BILD  SEHEN,=BILD
 ABLEGEN
470 DATA =HELP,BITTE WARTEN
480 FOR I=1 TO 9 :: READ Q$(
I):: NEXT I
490 ACCEPT AT(10,25)BEEP SIZ
E(-2)VALIDATE(DIGIT):SCR
500 IF SCR<3 OR SCR>15 THEN
490 ELSE CALL SCREEN(SCR)
510 ACCEPT AT(12,25)BEEP SIZ
E(-1)VALIDATE("OU"):WAHL$
520 IF WAHL$="O" THEN Z1=1 :
: Z2=16 :: Z5=17 ELSE Z1=9 :
```

**Listing für schnelle Zeichendefinition »Screen-Design« (Fortsetzung)**

```
: Z2=24 :: Z5=1
530 CALL CLEAR :: GOSUB 860
!---maske---
540 IF SPRUNG<>0 THEN GOSUB
940 ELSE SPRUNG=1
550 ! +----------------+
560 ! ! HAUPTPROGRAMM  !
570 ! ! zeichnen+menue !
580 ! +----------------+
590 FOR Z=Z1 TO Z2 :: FOR S=
1 TO 28
600 CALL HCHAR(Z,S+2,CHAR)
610 CALL TIC(CHAR,Z,S,K)
620 IF K=83 OR K=68 OR K=69
OR K=88 THEN GOSUB 790 :: GO
TO 600
630 IF K=48 THEN B(Z,S)=32
640 IF K=49 THEN GOSUB 890 :
: GOTO 600
650 IF K=50 THEN GOSUB 1330
:: GOSUB 860 :: GOTO 600
660 IF K=51 THEN GOSUB 970 :
: GOTO 600
670 IF K=52 THEN GOSUB 1220
:: GOTO 600
680 IF K=53 THEN GOSUB 920 :
: GOTO 520
690 IF K=54 THEN Z1=1 :: Z2=
24 :: GOSUB 940 :: CALL GET(
K):: CALL TITEL :: GOTO 490
700 IF K=55 THEN 1440
710 IF K=56 THEN GOSUB 1280
:: GOTO 600
720 IF K=13 THEN B(Z,S)=CHAR
 :: CALL HCHAR(Z,S+2,CHAR)
730 CALL HCHAR(Z,S+2,B(Z,S))
740 NEXT S :: NEXT Z :: GOTO
 590
750 !+--------------------+
760 ! ! ENDE HAUPTPROGRAMM  !
770 !+--------------------+
780 !--u.progr.cursor------
790 CALL HCHAR(Z,S+2,B(Z,S))
800 ON POS("SDXE",CHR$(K),1)
GOTO 810,920,830,840
810 S=S-1 :: IF S<>0 THEN RE
TURN ELSE S=28 :: RETURN
820 S=S+1 :: IF S<>29 THEN R
ETURN ELSE S=1 :: RETURN
830 Z=Z+1 :: IF Z<>Z2+1 THEN
 RETURN ELSE Z=Z1 :: RETURN
840 Z=Z-1 :: IF Z<>Z1-1 THEN
 RETURN ELSE Z=Z2 :: RETURN
850 !---maske anzeigen-----
860 K=0 :: FOR I=Z5 TO Z5+7
:: K=K+1 :: CALL HCHAR(I,11,
K+48):: CALL HCHAR(I,3,42,8)
:: DISPLAY AT(I,10):Q$(K)::
NEXT I
870 CALL FARBANZ(CV(),CH(),C
HAR,Z5):: RETURN
880 !------askii-holen-----
890 DISPLAY AT(Z5,10):"WELCH
ES ZEICHEN ?" :: ACCEPT AT(Z
5,27)BEEP SIZE(1):CHAR$ :: I
F CHAR$="" THEN 890
900 CHAR=ASC(CHAR$):: DISPLA
Y AT(Z5,10):Q$(1):: CALL FAR
BANZ(CV(),CH(),CHAR,Z5):: RE
TURN
910 !-------bild wechsel---
920 SPRUNG=1 :: IF WAHL$="O"
 THEN WAHL$="U" :: RETURN EL
SE WAHL$="O" :: RETURN
930 !--bild zeigen--
940 FOR Z=Z1 TO Z2 :: FOR S=
1 TO 28 :: CALL HCHAR(Z,S+2,
B(Z,S)):: NEXT S :: NEXT Z
950 SPRUNG=1 :: FOR I=140 TO
 110 STEP -10 :: CALL SOUND(
100,I,5,I+1,5):: NEXT I :: R
ETURN
960 !--char neu erstellen--
970 DISPLAY AT(Z5+2,9):"FELD
 LOESCHEN ? JN" :: CALL PAUS
E(100):: CALL GET(K):: IF K=
78 OR K=110 THEN 990
980 I=0 :: FOR K=Z5 TO Z5+7
:: I=I+1 :: A(I,1),A(I,2),A(
I,3),A(I,4),A(I,5),A(I,6),A(
I,7),A(I,8)=0 :: CALL HCHAR(
```

```
K,3,42,8):: NEXT K
990 DISPLAY AT(Z5+2,9):"1=EN
DE  0=LOESCHEN"
1000 FOR ZE=Z5 TO Z5+7 :: FO
R SP=1 TO 8
1010 ZEI=ZE :: IF ZEI>8 THEN
 ZEI=ZEI-16
1020 CALL TIC(30,ZE,SP,K)
1030 IF K=83 OR K=68 OR K=69
 OR K=88 THEN GOSUB 1080 ::
GOTO 1010
1040 IF K=49 THEN CALL HCHAR
(ZE,SP+2,42+A(ZEI,SP)):: GOT
O 1140
1050 IF K=48 THEN K=0 ELSE K
=1
1060 A(ZEI,SP)=K :: CALL HCH
AR(ZE,SP+2,42+K):: NEXT SP :
: NEXT ZE :: GOTO 1000
1070 !----u.progr.cursor----
1080 CALL HCHAR(ZE,2+SP,42+A
(ZEI,SP)):: ON POS("SDXE",CH
R$(K),1)GOTO 1090,1100,1110,
1120
1090 SP=SP-1 :: IF SP<>0 THE
N RETURN ELSE SP=8 :: RETURN
1100 SP=SP+1 :: IF SP<>9 THE
N RETURN ELSE SP=1 :: RETURN
1110 ZE=ZE+1 :: IF ZE<>Z5+8
THEN RETURN ELSE ZE=Z5 :: RE
TURN
1120 ZE=ZE-1 :: IF ZE<>Z5-1
THEN RETURN ELSE ZE=Z5+7 ::
RETURN
1130 !----auserechnen-------
1140 M$="" :: FOR R=1 TO 8 :
: RE=A(R,5)*8+A(R,6)*4+A(R,7
)*2+A(R,8)+1 :: LI=A(R,1)*8+
A(R,2)*4+A(R,3)*2+A(R,4)+1
1150 M$=M$&SEG$(HEX$,LI,1)&S
EG$(HEX$,RE,1):: NEXT R
1160 DISPLAY AT(Z5+2,9):"  Z
EICHENKODE ?"&STR$(CHAR+1)::
 ACCEPT AT(Z5+2,24)SIZE(-3)V
ALIDATE(DIGIT):X1
1170 IF (X1<33 OR X1>143)OR(
NOT(X1>47 OR X1<40))THEN 116
0
1180 IF ZEI$(X1)="" THEN NEU
$=NEU$&CHR$(X1)
1190 ZEI$(X1)=M$ :: CALL CHA
R(X1,M$):: CHAR=X1 :: DISPLA
Y AT(Z5+2,9):"3"&Q$(3)
1200 CALL FARBANZ(CV(),CH(),
CHAR,Z5):: RETURN
1210 !---color aendern-----
1220 DISPLAY AT(Z5+3,9):"CAL
L COLOR(   ,   ,   )"
1230 FOR I=1 TO 3 :: ACCEPT
AT(Z5+3,17+(I*3))VALIDATE(DI
GIT)SIZE(2):ZVH(I):: NEXT I
1240 IF ZVH(1)=2 OR ZVH(1)>1
4 OR ZVH(2)>16 OR ZVH(3)>16
THEN 1220
1250 CALL COLOR(ZVH(1),ZVH(2
),ZVH(3)):: CV(ZVH(1))=ZVH(2
):: CH(ZVH(1))=ZVH(3)
1260 DISPLAY AT(Z5+3,9):"4"&
Q$(4):: CALL FARBANZ(CV(),CH
(),CHAR,Z5):: RETURN
1270 !-------help----------
1280 RR=CHAR :: CALL FARBANZ
(CV(),CH(),RR,Z5):: GOTO 131
0
1290 CALL GET(K):: IF K=13 T
HEN CHAR=RR :: CALL FARBANZ(
CV(),CH(),RR,Z5):: RETURN
1300 IF K=69 AND RR<143 THEN
 RR=RR+1 ELSE IF K=88 AND RR
>33 THEN RR=RR-1
1310 DISPLAY AT(Z5+7,16):CHR
$(RR);RR :: GOTO 1290
1320 !-----char waehlen----
1330 CALL SPRITE(#1,47,2,(Z5
)*8,120) :: FOR I=Z5 TO Z5+7
:: DISPLAY AT(I,1):: NEXT I
:  DISPLAY AT(Z5+1,1):NEU$
1340 CALL MOTION(#1,0,0):: C
ALL GET(K):: ON 1+POS("SDXE"
&CHR$(13),CHR$(K),1)GOTO 134
0,1350,1360,1370,1380,1400
1350 X1=-4 :: Y1=0 :: GOTO 1
```

**Listing für schnelle Zeichendefinition »Screen-Design« (Schluß)**

```
390
1360 X1=4 :: Y1=0 :: GOTO 13
90
1370 X1=0 :: Y1=-4 :: GOTO 1
390
1380 X1=0 :: Y1=4
1390 CALL MOTION(#1,-10*Y1,1
0*X1):: GOTO 1340
1400 CALL POSITION(#1,X2,Y2)
:: X2=INT(X2/8+.5)+1 :: Y2=I
NT(Y2/8+.5)+1
1410 IF X2<1 OR X2>24 OR Y2<
1 OR Y2>32 THEN 1340 ELSE CA
LL GCHAR(X2,Y2,ZY)
1420 IF ZY=32 THEN CALL SOUN
D(100,110,0):: GOTO 1340 ELS
E CHAR=ZY :: CALL DELSPRITE(
#1):: RETURN
1430 !--basic-generator-----
1440 OPEN #1:"DSK1.SCREEN",D
ISPLAY ,SEQUENTIAL,OUTPUT,VA
RIABLE 163
1450 F,NR=0 :: GOSUB 1680 ::
 PRINT TAB(8);Q$(9):: CC$=CH
R$(157)&CHR$(200)
1460 Z$=Z$&ZI$&CC$&CHR$(5)&"
CLEAR"&CHR$(130)&CC$&CHR$(6)
&"SCREEN"&CHR$(183)&CHR$(200
)&CHR$(LEN(STR$(SCR)))&STR$(
SCR)&CHR$(182)
1470 GOSUB 1700 :: GOSUB 168
0
1480 Z$=Z$&ZI$&CC$&CHR$(5)&"
COLOR"&CHR$(183):: ZI$=""
1490 F=F+1 :: ZI$=ZI$&CHR$(2
00)&CHR$(LEN(STR$(F)))&STR$(
F)&CHR$(179)
1500 ZI$=ZI$&CHR$(200)&CHR$(
LEN(STR$(CV(F))))&STR$(CV(F)
)&CHR$(179)&CHR$(200)&CHR$(L
EN(STR$(CH(F))))&STR$(CH(F))
1510 IF F<>8 AND F<>14 THEN
ZI$=ZI$&CHR$(179):: GOTO 149
0 ELSE ZI$=ZI$&CHR$(182)
1520 Z$=Z$&ZI$ :: GOSUB 1700
1530 IF F<>14 THEN GOSUB 168
0 :: GOTO 1480
1540 GOSUB 1720 :: FOR I=1 T
O 6 :: GOSUB 1680 :: Z$=ZI$&
"A$"&CHR$(183)&CHR$(200)&CHR
$(1)&STR$(I)&CHR$(182)&CHR$(
190)&CHR$(199)
1550 Z$=Z$&CHR$(LEN(G$(I)))&
G$(I):: GOSUB 1700 :: NEXT I
 :: F=0
1560 CC$=CHR$(157)&CHR$(200)
&CHR$(4)&"CHAR"&CHR$(183)::
GOSUB 1680 :: Z$=ZI$&CC$
1570 FOR I=33 TO 143 :: IF L
EN(ZEI$(I))=0 THEN 1610 ELSE
 F=F+1
1580 Z$=Z$&CHR$(200)&CHR$(LE
N(STR$(I)))&STR$(I)&CHR$(179
)&CHR$(199)&CHR$(16)&ZEI$(I)
1590 IF I=143 OR F=LEN(NEU$)
THEN Z$=Z$&CHR$(182):: GOSUB
 1700 :: GOTO 1620
1600 IF F/5=INT(F/5)THEN Z$=
Z$&CHR$(182):: GOSUB 1700 ::
 GOSUB 1680 :: Z$=ZI$&CC$ EL
SE Z$=Z$&CHR$(179)
1610 NEXT I
1620 GOSUB 1680 :: Z$=ZI$&CH
R$(154)&CHR$(32)&"DISPLAY AT
(1,1):A$(1);A$(2);A$(3);A$(4
);A$(5);A$(6)" :: GOSUB 1700
1630 GOSUB 1680 :: Z$=ZI$&CH
R$(134)&CHR$(201)&ZI$ :: GOS
UB 1700
1640 !------close-----------
1650 Z$=CHR$(255)&CHR$(255):
: PRINT #1:Z$ :: CLOSE #1 ::
 CALL CLEAR
1660 PRINT "MIT MERGE DSK1.S
CREEN LADEN": :"ZEILENNUMMER
N :10-"&STR$(NR):: STOP
1670 !---zeilennummer------
1680 ZI$,Z$="" :: NR=NR+10 :
: ZI$=CHR$(INT(NR/256))&CHR$
(NR-256*INT(NR/256)):: RETUR
N
1690 !--zeile schliessen---
1700 Z$=Z$&CHR$(0):: PRINT #
```

```
1:Z$ :: RETURN
1710 !--bild in 6 strings--
1720 G$(1),G$(2),G$(3),G$(4)
,G$(5),G$(6)=""
1730 FOR Z=1 TO 24 :: X=INT(
(Z+3)/4):: FOR S=1 TO 28 ::
G$(X)=G$(X)&CHR$(B(Z,S)):: N
EXT S :: NEXT Z :: RETURN
1740 !
1750 !
1760 !
1770 !
1780 !
1790 !---------------------
1800 SUB GET(K)
1810 CALL KEY(0,K,S):: IF S=
0 THEN 1810
1820 SUBEND
1830 SUB TIC(CHAR,Z,S,K)
1840 CALL HCHAR(Z,S+2,32)::
GOSUB 1850 :: CALL HCHAR(Z,S
+2,CHAR):: GOSUB 1850 :: GOT
O 1840
1850 FOR I=1 TO 10 :: CALL K
EY(0,K,ST):: IF ST<>0 THEN S
UBEXIT
1860 NEXT I :: RETURN
1870 SUBEND
1880 SUB TITEL
1890 DISPLAY AT(1,8)ERASE AL
L:RPT$("*",13):TAB(7);"*";TA
B(21);"*":TAB(6);"* SCREEN D
ESIGN *":TAB(7);"*";TAB(21);
"*":TAB(8);RPT$("*",13)
1900 DISPLAY AT(10,2):"BILDS
CHIRMFARBE ? 3>15 4": :" OBE
N ODER UNTEN ?      U"
1910 SUBEND
1920 SUB FARBANZ(CV(),CH(),X
,Z):: P=INT(X/8)-3 ! P=ZEICH
ENSATZ
1930 DISPLAY AT(Z+7,16):CHR$
(X);X
1940 DISPLAY AT(Z+7,22):"V"&
STR$(CV(P))&" H"&STR$(CH(P))
:: SUBEND
1950 SUB PAUSE(X):: FOR I=1
TO X :: NEXT I :: SUBEND
```

```
1 ! DIESES PROGRAMM WURDE AB ZEILE 10 VOM PROGRAMM >SCREEN DESIGN< GENERIERT
2 ! ---------------------------------------------------------------------
10 CALL CLEAR :: CALL SCREEN(4)
20 CALL COLOR(1,2,1,2,2,16,3,2,1,4,2,1,5,2,1,6,2,1,7,2,1,8,2,1)
30 CALL COLOR(9,13,1,10,7,1,11,15,1,12,11,1,13,2,1,14,2,1)
40 A$(1)="                                                  qr                  qs
 x qpps                qpps       qpppps       "
50 A$(2)="  qs qpppps     qpppppps       qppppppppps  qppppppps     qppppppppppsq
pppppppppps   pppppppppppppppppppppppps  "
60 A$(3)="
                                            "
70 A$(4)="                                    jkl                          ih      j
kl                    b      ih             "
80 A$(5)="        gd      b                   edb       b                    gd
gd                   ab     edb             "
90 A$(6)="a  `aedb `a  agda   a   `   f`ff` fb`f `ffb f``af`f` `fff`aa`aacaa`aaa
ca`aaa`aa`aaaa`f``f```f``f`c``f``f``f``f`"
100 CALL CHAR(42,"0000000000000000",43,"FFFFFFFFFFFFFFFF",47,"FF818181818181FF",
96,"01818181A1212828",97,"0000404242525210")
110 CALL CHAR(98,"E0E0E0E0E0E0E0E0",99,"E0E0E0E8E8EAEAEA",100,"00F0F8F87C3C3E1F"
,101,"0003070F1E1C1C38",102,"0000000000042424")
120 CALL CHAR(103,"E0E3E7EFFFFEFEFC",104,"FFFFFEFEFCFCF8F0",105,"1F1F0F0F0707010
1",106,"80C0E0F87F7F3F3F",107,"000001071FFFFFFF")
130 CALL CHAR(108,"103060E0C0C08080",112,"FFFFFFFFFFFFFFFF",113,"0000030F1F3F7FF
F",114,"0000C0F0F8FCFEFF",115,"80C0E0F0F8F8FCFC")
140 CALL CHAR(120,"183C7EFFFF7E3C18")
150 REM DISPLAY AT(1,1):A$(1);A$(2);A$(3);A$(4);A$(5);A$(6)
160 GOTO 160
```

# Crazy Wood

## Finden Sie aus einem verrückten Zahlenwald heraus. Dies wird Ihnen aber nur mit List(ing) und Tücke gelingen.

Sie befinden sich im Zentrum eines nahezu undurchdringlichen Dickichts — grafisch dargestellt als buntes Zahlenlabyrinth. Da Ihre Vorräte nur begrenzt sind, und Sie unbedingt etwas trinken müssen, versuchen Sie den »verrückten Wald« zu verlassen. Wie Sie einer vergilbten Landkarte entnehmen konnten, gibt es vier schmale Pfade, die zurück in die Zivilisation führen, und zwar je einen im Norden, Westen, Süden und Osten.

Das Spiel beginnt in der Mitte des Waldes. An jedem Baum hängt ein Nummernschild. Die Zahl gibt an, um wieviele Schritte ein Spieler beim nächsten Zug vorrücken darf. Die Himmelsrichtung kann er dabei frei auswählen. Dadurch gelangt man zwar recht leicht an eine der vier Ausgangspforten, — hier beginnt jedoch die Schwierigkeit des Spiels. Der Spieler kann das Feld nämlich nur verlassen, wenn er genau so viele Schritte vorrücken darf, daß er exakt auf den Ausgangspunkt gelangt.

Das Programm beginnt mit einem vorbereiteten Teil (bis Zeile 790), in dem die »DIMENSION«-Anweisung, »DATA«-Zeilen, Character-Definitionen und ein vorgegebenes Spielfeld für das sogenannte Standardspiel abgelegt sind. Um einen schnelleren Programmstart zu ermöglichen, wurden die Zeilen 470 bis 1640 in Pre-Scan-Kommandos eingeschlossen.

Im Anschluß folgt die Erstellung des Titelbildes (Zeile 800 bis 890). Das Betätigen einer beliebigen Taste führt schließlich zum Spielmenü (Zeile 900 bis 1000), das von einer passenden Melodie untermalt wird. Folgende Optionen stehen zur Auswahl:
1. Spielanleitung (Zeile 1010 bis 1350), 2. Standardspiel (Zeile 1520), 3. Zufallsspiel (Zeile 1360) und 4. Ende des Spiels (Zeile 2430 bis 2450).

Die Wahl des Standardspiels liefert ein stets gleiches Spielfeld, das als String-Array X$(1)—X$(23) abgelegt wird. Dieses vorgegebene Feld eignet sich insbesondere zum Üben oder auch zur Demonstration.

Mit der Wahl des Zufallspiels dagegen wird ein rein zufällig erzeugter Aufbau erstellt. Diese Spielvariante zeichnet sich außerdem dadurch aus, daß neben den Zahlenfeldern auch Fragezeichenfelder existieren. Die Länge des kommenden Tagesmarsches hängt in diesen Fällen allein vom Zufall ab.

Sollten Sie einmal gänzlich die Orientierung verloren haben, können Sie jederzeit den Feuerknopf betätigen und das Spiel ist beendet.

(Hans-Jürgen Adler/wb)

```
100 ! **********************
110 ! *                    *
120 ! *    CRAZY  WOOD     *
130 ! *                    *
140 ! * ------------------ *
150 ! *                    *
160 ! *    Version 1.2     *
170 ! *                    *
180 ! **********************
190 !
200 !
210 ! TI-99/4A + Ext. Basic
220 !      + Joysticks
230 ! ====================
240 !
250 !     copyrights by
260 !   Harald Schwarz und
270 ! Hans-Juergen Adler '84
280 !
290 !     E&F-computing
300 !
310 ! Hans-Juergen Adler
320 ! Kurze Str. 2
330 ! 3392 Clausthal-Zell.
340 ! Tel. 05323/3332
350 !
360 !
370 DIM X$(23),C(2),C$(2)::
CALL CLEAR :: ON BREAK NEXT
380 DATA 4,139,6,175,2,156,4
,139,4,156,6,175,2,185,2,175
,0,0
390 DATA 4,139,8,175,8,156,8
,139
400 DATA 9,4,3,1,7,3,9,1,6,2
,13,2,5,2,15,2,5,1,17,2,4,2,
17,1,4,1,19,1,4,1,19,2
410 DATA 3,2,19,1,3,1,21,1,3
,1,21,2,3,1,21,1,3,2,19,1,4,
1,19,2
420 DATA 4,1,19,1,4,2,17,1,5
,1,17,2,5,2,15,2,6,2,13,2,7,
3,9,1,9,4,3,1,
430 GOTO 470 :: CALL SCREEN
:: CALL COLOR :: CALL SOUND
:: CALL JOYST
440 CALL CHAR :: CALL GCHAR
:: CALL MAGNIFY :: CALL DELS
PRITE :: CALL SPRITE
450 CALL KEY :: CALL CHARPAT
 :: CALL HCHAR :: CALL VCHAR
 :: RANDOMIZE
```

```
460 T1,T2,HI,Z1,T,J,N,F$,A$,
B$,B1$,K,S,I,KM,ZE,SP,P,K1,K
2,Q,PN :: !@P-
470 RANDOMIZE :: CALL MAGNIF
Y(2)
480 C(1)=128 :: C(2)=112 ::
HI=99999
490 F$="FFFFFFFFFFFFFFFF" ::
 A$="003844447C444444"
500 C$(1)="1818181818181818"
 :: C$(2)="000000FFFF"
510 CALL CHAR(104,F$,34,"007
E7E7E7E7E7E",96,"00384404180
44438",64,A$)
520 CALL CHAR(35,"0000007844
784040",36,"0000004444444438
",37,"0000007C10101010")
530 FOR N=49 TO 57 :: CALL C
HARPAT(N,B$):: CALL CHAR(79+
N,B$):: CALL CHAR(63+N,B$)::
 NEXT N
540 CALL CHARPAT(63,B$):: CA
LL CHAR(121,B$,137,B$)
550 CALL CHAR(91,"0044003844
447C44",92,"0044007C4444447C
",93,"004400444444438")
560 CALL CHAR(141,"3C4299A1A
199423C",142,"00FFFF",143,"F
88080F08384FC03")
570 X$(1)=RPT$(" ",11)&"hh@h
h"
580 X$(2)=RPT$(" ",8)&"hhhhq
"&CHR$(134)&"qhhhh"
590 X$(3)="        "&"hhht"&CH
R$(131)&"s"&CHR$(135)&"r"&CH
R$(130)&"q"&CHR$(133)&"rhhh"
600 X$(4)="      hh"&CHR$(128
)&"s"&CHR$(132)&"p"&CHR$(128
)&"p"&CHR$(128)&"t"&CHR$(128
)&"v"&CHR$(128)&"r"&CHR$(132
)&"hh"
610 X$(5)="     hh"&CHR$(131)
&"v"&CHR$(131)&"x"&CHR$(133)
&"v"&CHR$(132)&"s"&CHR$(132)
&"w"&CHR$(134)&"u"&CHR$(133)
&"w"&CHR$(132)&"hh"
620 X$(6)="     h"&CHR$(130)&
"v"&CHR$(129)&"x"&CHR$(135)&
"r"&CHR$(132)&"u"&CHR$(134)&
"s"&CHR$(136)&"p"&CHR$(135)&
"v"&CHR$(132)&"w"&CHR$(129)&
"h"
630 X$(7)="    hhp"&CHR$(131)
&"v"&CHR$(135)&"s"&CHR$(129)
&"x"&CHR$(129)&"v"&CHR$(128)
&"p"&CHR$(135)&"q"&CHR$(129)
&"v"&CHR$(133)&"rhh"
640 X$(8)="    hv"&CHR$(129)&
"p"&CHR$(135)&"t"&CHR$(132)&
"r"&CHR$(128)&"p"&CHR$(130)&
"p"&CHR$(130)&"r"&CHR$(131)&
"q"&CHR$(135)&"u"&CHR$(128)&
"rh"
650 X$(9)="    h"&CHR$(131)&"
q"&CHR$(133)&"v"&CHR$(129)&"
t"&CHR$(129)&"s"&CHR$(129)&"
q"&CHR$(132)&"p"&CHR$(130)&"
t"&CHR$(135)&"p"&CHR$(134)&"
v"&CHR$(130)&"h"
660 X$(10)="  hhs"&CHR$(128)
&"u"&CHR$(132)&"p"&CHR$(128)
&"p"&CHR$(136)&"p"&CHR$(131)
&"r"&CHR$(133)&"s"&CHR$(130)
&"p"&CHR$(136)&"w"&CHR$(129)
&"vhh"
670 X$(11)="  hs"&CHR$(130)&
"t"&CHR$(129)&"r"&CHR$(129)&
"q"&CHR$(130)&"q"&CHR$(131)&
"q"&CHR$(132)&"r"&CHR$(132)&
"p"&CHR$(128)&"r"&CHR$(132)&
"t"&CHR$(130)&"vh"
680 X$(12)="  @"&CHR$(130)&"
v"&CHR$(130)&"t"&CHR$(135)&"
p"&CHR$(130)&"v"&CHR$(132)&"
r`q"&CHR$(131)&"q"&CHR$(130)
&"v"&CHR$(134)&"t"&CHR$(131)
&"q"&CHR$(134)&"@"
690 X$(13)="  hp"&CHR$(132)&
"p"&CHR$(128)&"u"&CHR$(128)&
"q"&CHR$(131)&"s"&CHR$(133)&
"r"&CHR$(131)&"q"&CHR$(129)&
"p"&CHR$(129)&"u"&CHR$(132)&
"p"&CHR$(135)&"wh"
700 X$(14)="  hhs"&CHR$(130)
&"v"&CHR$(132)&"p"&CHR$(136)
&"r"&CHR$(131)&"s"&CHR$(132)
&"q"&CHR$(136)&"s"&CHR$(128)
&"x"&CHR$(132)&"v"&CHR$(131)
&"whh"
710 X$(15)="    h"&CHR$(131)&
"p"&CHR$(133)&"v"&CHR$(135)&
"r"&CHR$(131)&"r"&CHR$(131)&
"p"&CHR$(130)&"p"&CHR$(129)&
```

**Listing zu »Crazy Wood«**

**Listing zu »Crazy Wood« (Fortsetzung)**

```
"r"&CHR$(129)&"r"&CHR$(133)&
"q"&CHR$(131)&"h"
720 X$(16)="    hv"&CHR$(130)
&"q"&CHR$(133)&"p"&CHR$(132)
&"r"&CHR$(136)&"q"&CHR$(130)
&"q"&CHR$(129)&"t"&CHR$(134)
&"t"&CHR$(135)&"x"&CHR$(132)
&"sh"
730 X$(17)="    hhp"&CHR$(133
)&"v"&CHR$(130)&"s"&CHR$(135
)&"p"&CHR$(129)&"p"&CHR$(129
)&"p"&CHR$(129)&"q"&CHR$(135
)&"x"&CHR$(131)&"phh"
740 X$(18)="     h"&CHR$(129)
&"t"&CHR$(131)&"v"&CHR$(135)
&"v"&CHR$(132)&"u"&CHR$(128)
&"r"&CHR$(132)&"v"&CHR$(135)
&"v"&CHR$(129)&"x"&CHR$(130)
&"h"
750 X$(19)="     hh"&CHR$(133
)&"t"&CHR$(133)&"s"&CHR$(132
)&"v"&CHR$(129)&"t"&CHR$(129
)&"q"&CHR$(133)&"r"&CHR$(131
)&"v"&CHR$(131)&"hh"
760 X$(20)="      hh"&CHR$(12
9)&"u"&CHR$(128)&"t"&CHR$(13
0)&"t"&CHR$(130)&"s"&CHR$(13
2)&"r"&CHR$(129)&"t"&CHR$(12
9)&"hh"
770 X$(21)="       hhhv"&CHR$
(131)&"s"&CHR$(132)&"v"&CHR$
(130)&"s"&CHR$(131)&"vhhh"
780 X$(22)="         hhhhr"&C
HR$(130)&"shhhh"
790 X$(23)=RPT$(" ",11)&"hh@
hh"
800 REM
810 REM  Titelbild  ++++++++
820 N=1 :: CALL CLEAR :: CAL
L SCREEN(16)
830 FOR I=1 TO 14 :: CALL CO
LOR(I,7,16):: NEXT I :: CALL
 CHAR(140,C$(1))
840 CALL HCHAR(1,1,140,768)
850 DISPLAY AT(10,5)SIZE(20)
:" C R A Z Y  W O O D "
860 DISPLAY AT(23,1)SIZE(21)
:" "&CHR$(141)&" E&F-com##%i
ng '84 "
870 CALL KEY(0,K,S):: IF S<=
0 THEN N=N+1+(N>1)*2 ELSE 90
0
880 CALL CHAR(140,C$(N)):: G
OTO 870
890 GOTO 870
900 REM
910 REM  Spielmenue  +++++++
920 CALL CLEAR :: CALL SCREE
N(7):: FOR I=1 TO 14 :: CALL
 COLOR(I,2,11):: NEXT I
930 J=0 :: DISPLAY AT(3,5):"
BITTE WIHLEN SIE:":TAB(5);RP
T$(CHR$(142),17)
940 DISPLAY AT(12,5):"1  SPI
ELANLEITUNG": :"    2  STAND
ARDSPIEL"
950 DISPLAY AT(16,5):"3  ZUF
ALLSSPIEL": :"    4  ENDE" :
: N=0
960 RESTORE 380 :: FOR J=1 T
O 13 :: READ T1,T2
970 IF T1=0 THEN CALL SOUND(
20,30000,30)ELSE CALL SOUND(
T1*100,T2,5,T2*2,8)
980 CALL KEY(0,K,S):: IF S<>
0 AND(K<49)+(K>52)=0 THEN K=
K-48 :: GOTO 1000
990 NEXT J :: CALL SOUND(100
0,30000,30):: N=N+1 :: IF N=
4 THEN 800 ELSE 960
1000 ON K GOTO 1010,1520,136
0,2430
1010 REM
1020 REM  Spielanleitung  ++
+
1030 CALL CLEAR :: CALL SCRE
EN(5):: FOR I=1 TO 14 :: CAL
L COLOR(I,2,8):: NEXT I
1040 DISPLAY AT(2,7):"SPIELA
NLEITUNG":TAB(7);RPT$(CHR$(1
42),14)
1050 DISPLAY AT(5,1):"SIE BE
FINDEN SICH INMITTEN  EINES
DICHTEN WALDES UND"
1060 DISPLAY AT(7,1):"VERSUC
HEN VERZWEIFELT DEN  WALDRA
ND ZU ERREICHEN."
1070 DISPLAY AT(11,1):"ES GI
BT JEDOCH NUR 4 PFADE, DIE Z
URICK ANS HELLE TAGES-"
1080 DISPLAY AT(13,1):"LICHT
 FIHREN!": : :"HINWEISE IBER
 DIE WEGSTRECKE"
1090 DISPLAY AT(17,1):"GEBEN
```

```
 IHNEN DIE BÄUME DES   WALDE
S:  JEDER BAUM TRÄGT"
1100 DISPLAY AT(19,1):"EINE
ZAHL, DIE IHNEN ZEIGT, WELCH
E STRECKE SIE MIT DEM"
1110 DISPLAY AT(21,1):"KOMME
NDEN TAGESMARSCH BEWÄL-TIGEN
 WERDEN."
1120 DISPLAY AT(24,27):">>"
1130 CALL KEY(0,K,S):: IF S=
0 THEN 1130 ELSE CALL CLEAR
1140 DISPLAY AT(2,1):"SIE K\
NNEN IHRE MARSCHRICH- TUNG M
IT HILFE DES JOYSTICKS"
1150 DISPLAY AT(4,1):"NR. 1
FESTLEGEN, WOBEI SIE  STETS
EINE DER 4 HIMMELS-"
1160 DISPLAY AT(6,1):"RICHTU
NGEN WÄHLEN MÜSSEN."
1170 DISPLAY AT(9,1):"DIE ZU
RÜCKGELEGTE WEGSTRECKEWIRD E
BENSO ANGEZEIGT WIE"
1180 DISPLAY AT(11,1):"DIE L
ÄNGE DES KOMMENDEN TA- GESMA
RSCHES."
1190 DISPLAY AT(15,1):"SIE H
ABEN DIE WAHL ZWISCHEN ZWEI
SPIELM\GLICHKEITEN:"
1200 DISPLAY AT(18,1):" A) E
IN VORGEGEBENES SPIEL-     F
ELD (ZUM ÜBEN BZW. ZUR"
1210 DISPLAY AT(20,1):"     D
EMONSTRATION ODER": :" B) EI
N ZUFÄLLIG ERZEUGTES"
1220 DISPLAY AT(23,1):"     T
ERRAIN.":TAB(27);">>"
1230 CALL KEY(0,K,S):: IF S<
=0 THEN 1230 ELSE CALL CLEAR
1240 DISPLAY AT(2,1):"DER ZU
FÄLLIG ERZEUGTE SPIEL-AUFBAU
 'B' WEIST EINE BESON-"
1250 DISPLAY AT(4,1):"DERHEI
T GEGENÜBER DEM STAN- DARDSP
IEL AUF: AUSSER DEN"
1260 DISPLAY AT(6,1):"ZAHLEN
FELDERN, WELCHE DIE   LÄNGE
DES NÄCHSTEN TAGES-"
1270 DISPLAY AT(8,1):"MARSCH
ES ANZEIGEN, KOMMEN   AUCH V
EREINZELT FELDER VOR,"
1280 DISPLAY AT(10,1):"DIE E
IN FRAGEZEICHEN AUFWEI-SEN.
IN DIESEN FÄLLEN HÄNGT"
1290 DISPLAY AT(12,1):"DER K
OMMENDE TAGESMARSCH    ALLEI
N VOM ZUFALL AB."
1300 DISPLAY AT(15,1):"**  S
OLLTEN SIE SICH DER- ****  A
RT VERLAUFEN HABEN,  **"
1310 DISPLAY AT(17,1):"**  D
ASS DER HUNGERTOD    ****  D
ROHT, SO K\NNEN SIE  **"
1320 DISPLAY AT(19,1):"**  J
EDERZEIT MIT HILFE   ****  D
ES 'FIRE-BUTTONS'    **"
1330 DISPLAY AT(21,1):"**  A
UFGEBEN !            **"
1340 DISPLAY AT(23,6):"V I E
 L   S P A S S"
1350 CALL KEY(0,K,S):: IF S=
0 THEN 1350 ELSE 900
1360 REM
1370 REM  Neues Spielfeld  +
+
1380 RESTORE 400 :: CALL CLE
AR :: CALL SCREEN(4)
1390 CALL COLOR(1,8,13,2,2,1
3,3,2,13,4,2,13,6,2,13,7,2,1
3)
1400 CALL COLOR(5,16,14,9,2,
16,10,2,11,11,2,12,12,2,12,1
3,16,7,14,16,7)
1410 DISPLAY AT(1,1):X$(1)
1420 FOR I=2 TO 22 :: READ N
,K,S,T
1430 DISPLAY AT(I,N):RPT$("h
",K);TAB(N+K+S);RPT$("h",K)
1440 B$="" :: FOR J=1 TO S :
: K1=INT(RND*10)+C(T):: B$=B
$&CHR$(K1)
1450 T=T+1+(T>1)*2
1460 NEXT J :: DISPLAY AT(I,
N+K)SIZE(S):B$ :: IF I<>12 T
HEN 1510
1470 DISPLAY AT(I,3)SIZE(1):
"@" :: DISPLAY AT(I,25):"@"
1480 CALL HCHAR(12,16,121)
1490 J=0 :: CALL GCHAR(12,16
,Z1):: IF Z1=121 THEN Z1=63
:: J=1 ELSE Z1=Z1-63
1500 CALL CHARPAT(Z1,B1$)::
CALL CHAR(97+J,B1$):: CALL H
CHAR(12,16,97+J)
1510 NEXT I :: DISPLAY AT(23
```

Listing zu »Crazy Wood« (Fortsetzung)

```
,1):X$(23):: GOTO 1580
1520 REM
1530 REM  Spielfeld  +++++++
1540 CALL CLEAR :: CALL SCRE
EN(5)
1550 CALL COLOR(1,5,8,2,2,8,
3,2,8,4,2,8,6,2,8,7,2,8)
1560 CALL COLOR(5,2,12,9,2,4
,10,2,8,11,2,16,12,2,16,13,1
6,7,14,16,7)
1570 FOR I=1 TO 23 :: DISPLA
Y AT(I,1):X$(I):: NEXT I
1580 CALL SPRITE(#2,143,16+(
T<>2)*11,5,14)
1590 REM
1600 KM=0 :: ZE=12 :: SP=16
1610 CALL GCHAR(ZE,SP,P)
1620 DISPLAY AT(21,26)BEEP:"
POS"
1630 IF Q>0 THEN CALL SOUND(
-100,1400,0)
1640 CALL SPRITE(#1,P,2,176,
222)
1650 FOR I=1 TO 100 :: NEXT
I
1660 CALL JOYST(1,K1,K2):: C
ALL KEY(1,K,S)
1670 IF ABS(K1)+ABS(K2)+(K=1
8)=0 THEN 1660
1680 IF K=18 THEN CALL MEL :
: CALL CLEAR :: CALL DELSPRI
TE(#1,#2):: GOTO 900
1690 GOSUB 2290
1700 IF K2=4 THEN 1740
1710 IF K1=-4 THEN 1840
1720 IF K2=-4 THEN 1940
1730 IF K1=4 THEN 2040
1740 REM
1750 REM  Nord  ************
1760 IF ZE-Q<1 THEN 1660
1770 CALL VCHAR(ZE,SP,P)
1780 ZE=ZE-Q :: PN=P
1790 CALL GCHAR(ZE,SP,P)
1800 IF P=32 OR P=104 THEN Z
E=ZE+Q :: P=PN :: Q=0
1810 CALL VCHAR(ZE,SP,34)
1820 GOSUB 2380
1830 GOTO 1630
1840 REM
1850 REM  West  ************
1860 IF SP-Q<5 THEN 1660
1870 CALL VCHAR(ZE,SP,P)
1880 SP=SP-Q :: PN=P
1890 CALL GCHAR(ZE,SP,P)
1900 IF P=32 OR P=104 THEN S
P=SP+Q :: P=PN :: Q=0
1910 CALL VCHAR(ZE,SP,34)
1920 GOSUB 2380
1930 GOTO 1630
1940 REM
1950 REM  Sued  ************
1960 IF ZE+Q>23 THEN 1660
1970 CALL VCHAR(ZE,SP,P)
1980 ZE=ZE+Q :: PN=P
1990 CALL GCHAR(ZE,SP,P)
2000 IF P=32 OR P=104 THEN Z
E=ZE-Q :: P=PN :: Q=0
2010 CALL VCHAR(ZE,SP,34)
2020 GOSUB 2380
2030 GOTO 1630
2040 REM
2050 REM  Ost  *************
2060 IF SP+Q>27 THEN 1660
2070 CALL VCHAR(ZE,SP,P)
2080 SP=SP+Q :: PN=P
2090 CALL GCHAR(ZE,SP,P)
2100 IF P=32 OR P=104 THEN S
P=SP-Q :: P=PN :: Q=0
2110 CALL VCHAR(ZE,SP,34)
2120 GOSUB 2380
2130 GOTO 1630
2140 REM
2150 REM  Angekommen  ******
2160 CALL CLEAR :: CALL DELS
PRITE(#1,#2)
2170 CALL SCREEN(12):: FOR I
=1 TO 14 :: CALL COLOR(I,16,
5):: NEXT I
2180 FOR I=110 TO 440 STEP 1
10 :: FOR J=1 TO 4
2190 CALL SOUND(-100,I+110,0
):: CALL SOUND(-100,I+220,0)
:: NEXT J :: NEXT I
2200 DISPLAY AT(5,2):"HERZLI
CHEN GLJCKWUNSCH !!"
2210 DISPLAY AT(8,3):"SIE HA
BEN ES GESCHAFFT!"
2220 DISPLAY AT(14,2):"SIE H
ABEN EINE WEGSTRECKE": :" VO
N";KM;"KM ZURJCKGELEGT."
2230 IF KM>HI THEN 2270 ELSE
 HI=KM
2240 FOR N=1 TO 5 :: DISPLAY
 AT(20,2):"BEST-SCORE:";HI;"
```

**Listing zu »Crazy Wood« (Schluß)**

```
KM"
2250 CALL SOUND(150,220,5,44
0,5,660,5):: DISPLAY AT(20,1
)
2260 CALL SOUND(150,30000,30
):: NEXT N
2270 DISPLAY AT(20,2):"BEST-
SCORE:";HI;"KM" :: CALL SOUN
D(150,220,5,440,5,660,5)
2280 CALL KEY(0,K,S):: CALL
KEY(1,K1,K2):: IF S<=0 AND K
2<=0 THEN 2280 ELSE 900
2290 REM
2300 REM  Wegstrecke kontrol
lieren  **********
2310 IF P>127 THEN Q=P-127
2320 IF P>111 AND P<128 THEN
 Q=P-111
2330 IF P=96 THEN Q=3 :: RET
URN
2340 IF P=97 THEN Q=Z1-48 ::
 RETURN
2350 IF P=98 OR P=121 OR P=1
37 THEN Q=INT(RND*9)+1 :: RE
TURN
2360 IF P=64 THEN 2140
2370 RETURN
2380 REM
2390 REM  km-Zaehler  ******
2400 KM=KM+Q
2410 DISPLAY AT(24,1):USING
"##### KM":KM
2420 RETURN
2430 REM
2440 REM  Ende  ++++++++++++
2450 CALL CLEAR :: END
2460 !@P+
2470 REM
2480 SUB MEL
2490 FOR I=880 TO 110 STEP -
55
2500 CALL SOUND(-200,I+110,0
):: CALL SOUND(-300,I,3):: N
EXT I :: SUBEND
```

# Schlüssel, Schlangen, Schätze

**Daß man selbst auf der winzigen Anzeige des Sharp PC-1245 mit Grafikzeichen arbeiten kann, zeigen wir mit diesem netten Spielchen. Es geht um dunkle Gänge, gefräßige Monster und einen Schatz.**

Zunächst erscheint der Name des Spiels, »Key«, in leicht verfremdeten Buchstaben, danach die Anweisung »OPEN 7 DOORS«. Nun befinden Sie sich im ersten von sieben Räumen. Links auf dem Display sehen Sie ein Schlüsselloch, daneben das Männchen, das Sie über das Spielfeld bewegen müssen. Rechts wird das Feld von einem Schlüssel begrenzt. In der Mitte wartet ein Monster — im ersten Bild ist es eine Schlange.

Ziel des Spiels ist es, heil am Monster vorbeizukommen, den Schlüssel zu holen und wieder zurück zum Schlüsselloch zu gelangen. Am Anfang haben Sie fünf Männchen zur Verfügung. Die verbleibende Anzahl wird rechts neben dem Schlüssel in Strichen angezeigt. Das Männchen bewegen Sie mit der Taste »4« nach links, mit »6« nach rechts.

Am Monster kommen Sie folgendermaßen vorbei: Das Männchen muß mit ihm »Feld an Feld« stehen. Dann muß man abschätzen, wann das Monster sich auf dessen Platz bewegt. In diesem Moment muß das Männchen in die entgegengesetzte Richtung springen. Es hat dann praktisch den Platz getauscht. Man sollte dieses Spiel nur mit starken Batterien spielen, da sonst die Anzeige zu langsam reagiert, und sich das Spiel zum »Glücksspiel« entwickelt. Ist das Männchen jetzt am Feld vor dem Schlüssel angelangt, fängt dieser zu blinken an: Man hat ihn »eingesteckt«. Jetzt muß man wieder am Monster vorbei und zurück zum Ausgangspunkt. Dann verschwindet

Anmerkung
Wir haben dieses Programm auf dem PC-1251 getestet. Dabei traten im äußersten linken und rechten Display-Segment einige merkwürdige Erscheinungen auf. Den Spielablauf beeinflußten sie jedoch nicht, soweit wir feststellen konnten.

# Noch mehr Joy ohne Stick

## Die Simulation eines Joysticks auf der TI-Tastatur hat ein großes Echo unter unseren Lesern gefunden.

Viele Leser fanden die Joystick-Routine, die wir in Happy Computer 7 unter dem Titel »Auch ohne Stick viel Joy« vorstellten, viel zu träge und umständlich. Etliche machten sich daran, das kurze Listing zu verbessern. Lesen Sie drei wirkungsvolle Lösungen, die uns die Leser zugeschickt haben.

### Vorschlag 1

Die abgedruckte Routine benötigt acht Tasten um sämtliche Joystick-Rückmeldungen zu simulieren. Einfacher ist es, nur vier Tasten zu belegen. Mann kann die Abfrage links, rechts beziehungsweise nach oben und unten, inklusive der Diagonalwerte durch Mischen erzeugen. In ähnlicher Weise funktioniert auch ein Joystick.

Durch einfaches Umschalten der Tastaturmodi 1 und 2 kann die Tastatur des TI 99/4A geteilt werden. Ich benutze dazu die Tasten »Z« für links, »X« für rechts, »;« für unten und »/« für oben. Das Kommando »links oben« wird dann durch gleichzeitiges Betätigen der »Z« und »/« Tasten bewerkstelligt. Diese Lösung ist sicherlich einfacher, als wenn man mit acht Tasten arbeiten müßte.

Speziell bei Spielen kommt es auf eine möglichst schnelle Joystick-Abfrage an. Das in Ausgabe 7 abgedruckte Programm dürfte für schnelle Spiele zu langsam sein. Mehrfache »IF THEN«-Abfragen sollte man deshalb vermeiden. Schnell hintereinander auszuführende Befehle sollte man des weiteren in einer oder möglichst wenigen Programmzeilen unterbringen.

Tatsächlich kann man in Extended Basic das gesamte Unterprogramm zur Joystick-Abfrage in einer einzigen Programmzeile unterbringen. Das sähe dann so aus:

```
5000 SUB JOYST(1,X,Y)::CALL KEY(1,K,S)::X=4*((K=15)-(K=0))::CALL KEY(2,K,S)::Y=4*((K=17)-(K=16))::SUBEND
```

Man muß dann in der obigen Programmzeile, bei jedem Aufruf das »O« des »CALL JOYST(1,X,Y)« in eine Null umwandeln.

So wie die Programmzeile oben abgedruckt ist, leistet sie in etwa so viel wie die Joystickabfrage aus der Happy-Computer Ausgabe 7. Es soll nicht verschwiegen werden, daß das von mir vorgestellte Unterprogramm um etliches schneller ist.

Zu einem guten Programmierstil gehört weiterhin, daß man entweder die Tastatur oder die Joysticks als Eingabemedium verwendet. Am einfachsten ist es, wenn man zum Programmstart nach der gewünschten Art der Steuerung fragt. Die gewählte Option könnte dann in einer Variablen »JOY« abgespeichert werden, die normalerweise den Wert »0« enthält. Bevorzugt man die Fernbedienung, könnte die Variable den Wert »-1« annehmen. Die entsprechende Programmänderung könnte dann so aussehen:

```
10 DISPLAY AT(23,1)BEEP:"WILLST DU EINEN JOYSTICK":"VERWENDEN?"
11 CALL KEY(3,K,S)::IF S=0 THEN 11 ELSE JOY=(K=74)
..
..
5000 SUB JOYST(JOY,X,Y)::IF JOY THEN CALL JOYST(1,X,Y)::SUBEXIT
5010 CALL KEY(1,K,S)::X=4*((K=15)-(K=0))::CALL KEY(2,K,S)::Y=4*((K=17)-(K=16))::SUBEND
```

So ist das Programm schon wesentlich komfortabler, aber immer noch nicht perfekt. Ein gutes Programm sollte benutzerfreundlich sein. Bekanntlich ist der TI 99/4A für den Anschluß von zwei Joysticks ausgerüstet. Jetzt könnte man es natürlich dem Benutzer überlassen welchen Joystick er verwenden möchte. Ich verfahre dann folgendermaßen: Durch Betätigung des entsprechenden Feuerknopfes kann der Benutzer selber wählen welchen Joystick er verwenden möchte.

```
10 DISPLAY AT(23,1)BEEP:"WILLST DU EINEN JOYSTICK":"VERWENDEN?"
11 CALL KEY(3,K,S)::IF S=0 THEN 11 ELSE JOY=(K=74)
12 IF NOT JOY THEN 100 !SPRUNG INS HAUPTPROGRAMM
13 DISPLAY AT(24,1)ERASE ALL BEEP:"STARTE MIT DER FEUERTASTE"
14 FOR NR=1 TO 2
15 CALL KEY(NR,K,S)::IF K=18 THEN 100
16 NEXT NR::GOTO 14
..
..
100 REM HAUPTPROGRAMM
..
..
5000 SUB JOYST(JOY,NR,X,Y)::IF JOY THEN CALL JOYST(NR,X,Y)::SUBEXIT
5010 CALL KEY(1,K,S)::X=4*((K=15)-(K=0))::CALL KEY(2,K,S)::Y=4*((K=17)-(K=16))::SUBEND
```

Die entsprechenden Aufrufe »CALL JOYST(1,X,Y)« beziehungsweise »CALL JOYST(2,X,Y)« des umzuarbeitenden Programms müssen dann durch »CALL JOYST(JOY,NR,X,Y)« ersetzt werden.

Weniger erfahrene Programmierer könnten die Abfragen wie »IF JOY THEN ..« oder »IF NOT JOY THEN ..« vielleicht falsch verstehen. Es handelt sich hierbei um verkürzte, und wie ich finde, einprägsame Schreibweisen für »IF JOY<>0 THEN ..« und »IF JOY=0 THEN ..«.

Den in Klammern gesetzten Gleichungen in Zeile 5010 sind Zahlenwerte zugeordnet. Wie das funktioniert wird auf den Seiten 56 und 59 des TI Basic-Handbuches erklärt. Leider haben nur wenige TI 99/4A-Besitzer die Möglichkeiten genutzt die »LET« und »IF THEN ELSE« Befehle zu umschreiben. Dabei sind diese Möglichkeiten wichtig, wenn es auf Schnelligkeit und möglichst wenig Speicherplatz ankommt.

Hier noch eine Anmerkung für Benutzer, denen das Extended-Basic-Modul nicht zur Verfügung steht: Die oben erwähnten Abfragen funktionieren übrigens nicht vollständig auf der Grundversion des TI 99/4A. Für die links und rechts Abfrage können die »Z«- und »X«-Tasten nicht verwendet werden, da es hier zu einem Fehler kommt. Wahrscheinlich liegt hier ein Mangel im Betriebssystem vor, da »K=0« nicht der geforderte Wert »-1« zugeordnet wird, wenn man die »X«-Taste betätigt. Ich verwende in diesem Fall die »A« und »S« Tasten, die auch ganz bequem mit der linken und rechten Hand zu erreichen sind. Das Basic Programm sieht dann so aus:

# TI 99/4A

```
5000 IF JOY THEN 5060
5010 CALL KEY(1,K,S)
5020 X=4*((K=1)-(K=2))
5030 CALL KEY(2,K,S)
5040 Y=4*((K=17)-(K=16))
5050 RETURN
5060 CALL JOYST(NR,X,Y)
5070 RETURN
```

Man muß nur beachten, daß jeder »CALL JOYST(...)« Aufruf im Programm durch »GOSUB 5000« ersetzt wird.

(Dieter Taube/wb)

## Vorschlag 2

Der folgende Vorschlag setzt die Konsole sowie das Extended Basic Modul voraus:

```
30010 SUB JOYSTICK(X,Y)
30011 X,Y=0::CALL KEY(0,K,S)::IF S=0 THEN SUBEXIT
30012 X=POS("ZSWzswCDRcdr",CHR$(K),1)::IF X THEN X=4*SGN(X-6.5)
30013 Y=POS("ZXCzxcWERwer",CHR$(K),1)::IF Y THEN Y=4*SGN(Y-6.5)
30014 SUBEND
```

Die obige Kurzroutine simuliert einen Joystick auf der Tastatur und arbeitet verhältnismäßig schnell. Die Stellung der »ALPHA LOCK«-Taste ist dabei unwichtig. Eine Anpassung an TI-Basic kann durch weglassen der »SUB«-Befehle und hintereinander schreiben der restlichen Programmanweisungen leicht vorgenommen werden. »SUBEXIT« und »SUBEND« sind dann durch »RETURN« zu ersetzen.

(Stefan Schwarzer/wb)

## Vorschlag 3

Die meisten »IF«-Abfragen lassen sich mit nur geringen Kenntnissen in Mathematik und etwas logischem Denkvermögen vermeiden. Hier mein Vorschlag:

```
30000 SUB JOYST(N,X,Y)
30010 CALL KEY(N,K,S)::X=((K=2)+(K=4)+(K=15)-(K=3)-(K=6)-(K=14))*4::Y=((K=0)+(K=15)+(K=14)-(K=5)-(K=4)-(K=6))*4
30020 SUBEND
```

Mit Hilfe der »REDO«-Taste ist es dann möglich das Unterprogramm in eine Zeile zu schreiben. Versieht man die Zeile 30000 mit einem »REM« und schreibt jeden Befehl in eine Zeile, kann das Unterprogramm auch von TI-Basic aus aufgerufen werden. In diesem Fall müßte man es mit »GOSUB 30000« aufrufen. Allerdings muß vorher der Tastaturmodus mit »N=1« oder »N=2« angewählt werden.

Diese Möglichkeit steht einem mit Extended Basic auch zur Verfügung. Es sind dann allerdings die Werte für K in Zeile 30010 an die entsprechenden ASCII-Werte anzupassen. Auf diese Art und Weise lassen sich dann beliebige Tasten zur Steuerung verwenden.

In Extended Basic erfolgt der Aufruf mit »CALL JOYST(N,X,Y)«, entsprechend dem Aufruf der Joysticks, wenn diese angeschlossen sind.

Sollten Joysticks angeschlossen sein, wird dennoch nur das Unterprogramm für die Tastatursteuerung ausgeführt. Möchte man sie aber verwenden, ist lediglich das Unterprogramm zu entfernen. Das hat den Vorteil, daß nicht zu viele Änderungen im Programm vorgenommen werden müssen.

(Dietrich Terasa/wb)

# IDEENECKE

**VC 20 als Schaltuhr und Terminkalender:** Programm gesucht, das den VC 20 in eine Schaltuhr, zum Beispiel für einen Kaffeeautomaten verwandelt oder akustisch an Termine mit gleichzeitiger Textausgabe erinnert. Außerdem wäre eine ständige Zeit- und Datumsanzeige erwünscht.

Michael Bilger, Alb-Braun-Str. 10g, 7500 Karlsruhe 21

**Hubschrauber-Flugsimulator für VC 20:** Der Flugsimulator müßte genauso detailgetreu sein, wie die bekannten Simulator-Programme für Sportflugzeuge, also mit Höhen- und Geschwindigkeitsmesser, Tankanzeige, künstlichem Horizont, Flight Director und Uhr. Eine Möglichkeit für Nachtflugsimulation wäre schön.

Stefan Scheuer, Boelckestr. 4, 5014 Kerpen

**Asterix auf Computer:** Grundidee für ein Asterix-Abenteuer auf dem Computer könnte die Geschichte »Asterix, die Odyssee« sein. Dem Druiden ist das wichtige Steinöl ausgegangen und die beiden Gallier, Asterix und Obelix, müssen dieses Öl aus Mesopotamien holen. Dabei haben sie viele Abenteuer zu bestehen.

Stefan Gallerdörfer, Roter Weiher 27, 8480 Weiden 6

**Regatta-Plazierung auf dem Commodore 64 oder HX20:** Ich suche ein Programm, das bei einer Regatta nach den Vorläufen die Hauptgruppen festlegt und nach jedem Hauptlauf die aktuelle Plazierung anzeigt. Das Problem ist, daß nach jedem Lauf die zwei besten aufsteigen und die zwei schlechtesten absteigen.

Peter Warwel, Am Kirschenberg 5, 6365 Rosbach 1

**Schiff-Simulator für Commodore 64:** Es müßte möglich sein, ein Schiff über Joystick und Tastatur zu steuern. Während des Spiels soll sich das Wetter und der Seegang ständig verändern. Man müßte außerdem unterschiedliche Häfen anlaufen können.

Sven Pioch, Neue Str. 16, 1000 Berlin 42

**Unendliche Geschichte für VC 20 mit maximal 20 KByte:** Ein Grafik-Adventure nach dem Stoff der Unendlichen Geschichte, vor allem den Abenteuern des Atrejus würde sicher viele Interessenten finden.

Robin Kositza, John-F.-Kennedy-Allee 39, 3180 Wolfsburg 1, Tel. 05361/7 25 87

**Manch eine gute Programm-Idee wartet ein Leben lang darauf, ausgeführt zu werden...**
**...und mancher Programmierer wartet ein Leben lang auf eine gute Idee. Wir wollen beide zusammenbringen.**

### Wer eine Idee hat

★ schickt uns diese Idee, mit oder ohne nähere Erläuterung, auf einer Karte oder in einem Brief, mit Namen und Adresse versehen

### Wer eine Idee aufgreift

★ und in ein Programm einbaut, erwähnt neben dem eigenen Namen und der eigenen Adresse den Namen und die Adresse des Ideenanbieters in der Kopfzeile des Programms oder der entsprechenden Subroutine.

**Einsendungen an**
**Happy Computer**
**Aktion Ideenecke**
**Hans-Pinsel-Str. 2**
**8013 Haar bei München**

# Flugsimulator

**Der Flugsimulator ist ein spannendes Spiel für den TI99/4A mit Extended-Basic-Erweiterung. Er simuliert bei farbiger Grafik den Flug eines Flugzeuges und verlangt vom Piloten einiges an Konzentration und Geschicklichkeit.**

Bei diesem Flugsimulator sitzt man nicht wie üblich im Flugzeug, sondern steuert das Flugzeug von außen. Die Bewegung der Maschine wird einfach durch vorbeiziehende Hintergrundsprites vorgetäuscht.

Ziel des Spieles ist, das Flugzeug zu starten, zum nächsten Flughafen zu fliegen und dort wieder sicher zu landen oder, falls erforderlich, notzulanden. Bis man aber soweit ist, muß man sich erst mit dem Flugzeug und den »Flugregeln« vertraut machen, die in der Tabelle zusammengefaßt sind.

| Taste | Funktion |
|---|---|
| Z | beschleunigen |
| . | abbremsen |
| E | steigen und abheben |
| X | sinken und landen |
| 1 | Flugzeug in Rollstellung |
| 2 | Flugzeug in Schrägstellung |
| 3 | Flugzeug in Flugstellung (Fahrwerk eingefahren) |
| S | Flug nach links (nur für Landungsradar) |
| D | Flug nach rechts (nur für Landungsradar) |
| 0 | Auslösen des Schleudersitzes |
| F | Öffnen des Schleudersitzfallschirms |

**Tabelle. Mit diesen Tasten steuert man das Flugzeug**

Die Tasten sind so belegt, wie man sie sich am besten merken kann und die Bedienung des Flugzeugs am einfachsten ist. Sollte jemand mit dieser Anordnung nicht zurecht kommen, muß er nur die Werte der Rückgabevariablen »K« in den bei den »CALL KEY«-Befehlen stehenden Vergleichsoperationen ändern.

## Exakte Steuerung für guten Flug

**Starten:** Bei einer Geschwindigkeit (SPEED) größer gleich 190 kann das Flugzeug schräg gestellt werden. Zum Abheben muß das Flugzeug schräg stehen und die Geschwindigkeit größer gleich 200 sein.

| Flug: | |
|---|---|
| *Höhe* | *Richtwerte* |
| 10- 100 | Fahrwerk muß ausgefahren sein |
| 110- 200 | in diesem Abschnitt wird das Fahrwerk ein- beziehungsweise ausgefahren |
| 10- 600 | Geschwindigkeit höchstens 300 |
| 600- 700 | Geschwindigkeit größer 200 (Beschleunigungszone) |
| 700-1080 | Geschwindigkeit mindestens 400 |

**Landeanflug und Landung:** Das Fahrwerk darf erst ausgefahren werden, wenn der Flughafen nur noch 1000 Einheiten entfernt ist. Wenn die Ausrollgeschwindigkeit kleiner als 175 ist, muß das Flugzeug in Rollstellung gebracht worden sein. Der Flug gilt als beendet, wenn die Geschwindigkeit gleich 0 und die Landung erfolgreich verlaufen ist.

**Absprung mit dem Schleudersitz:** Diese Funktion ist nur für den äußersten Notfall vorgesehen, zum Beispiel wenn die Notlandebahn verfehlt wurde, oder das Flugzeug über die Landebahn hinauszuschießen droht. Mit der »0«-Taste sprengt man die Pilotenkanzel aus dem Flugzeug. Wenn sie sich im freien Fall befindet, muß man mit der »F«-Taste den Fallschirm öffnen und mit den Cursortasten »S« und »D« die Kanzel auf den schwarzen Strich am Grund zusteuern, um auf ihm zu landen.

**Landungsradar:** Kurz vor Erreichen des Flughafens beginnt mit dem Erscheinen des Landungsradars der Landeanflug. Mit den Tasten »S« und »D« wird das Flugzeug auf die Landebahn ausgerichtet. Dies ist dann erreicht, wenn der Radarpunkt auf dem schwarzen Strich des Radarschirms liegt.

**Durchstarten:** Falls man die Landung nicht durchführen kann, weil zum Beispiel die Flughöhe zu groß ist, startet man durch Drücken der »Z«-Taste durch. Allerdings sollte man seinen Treibstoffrest und die Entfernung zum nächsten Flughafen berücksichtigen, bevor man eine Entscheidung fällt. Die Entfernung zum nächsten Flughafen wird beim Landeanflug eingeblendet.

**Hindernisse:** Die einzelnen Hindernisse bedürfen keiner weiteren Erläuterung. Der Pilot muß in den einzelnen Fällen selbständig handeln.

**Radarschirm:** Wenn ein Flugobjekt auf Gegenkurs ist, erscheint der Radarschirm. Wenn der Radarpunkt auf dem schwarzen Feld liegt, besteht Kollisionsgefahr.

## Programm in Blöcken aufgeteilt

Die wichtigsten Abschnitte des Programms sind durch Remark-Zeilen gekennzeichnet und erläutert. So wird der Ablauf übersichtlich.

**Starten und Flug:** Zeile 440 wird nur beim Start (STS=0) beachtet, sonst verzweigt das Programm gleich zu Zeile 450. Die Zeilen 480 bis 510 werden beim Starten ebenfalls übersprungen. Die Zeilen 610 bis 620 werden nur beim Landeanflug (PPP=1) beachtet. Mit ihnen wird der Radarpunkt bewegt.

**Notlandung:** Die Zeilen 840 bis 910 werden benutzt, wenn das Flugzeug noch in der Luft ist. Sobald das Flugzeug Bodenkontakt hat, wird zu Zeile 930 gesprungen.

**Erzeugung der Hindernisse:** Hindernisse werden nur erzeugt, wenn der Schwierigkeitsgrad für Fortgeschrittene (SCHW=1) gewählt und wenn die Entfernung zum Flughafen größer als 7000 Einheiten ist. Welches Hindernis erzeugt wird, bleibt dem Zufall überlassen (Zeile 1360):

Zeile 1370 — Nebel
Zeile 1410 — Notlandung
Zeile 1440 — Berge
Zeile 1490 — Flugobjekt auf Gegenkurs
Zeile 1520 — Nebel löst sich auf

**Meldung von schweren Fehlern:** Sie führen zur Explosion des Flugzeugs (Zeile 800). Je nach Fehler springt der Computer zu Zeile 300 und ein neuer Versuch muß gestartet werden.

**Meldung von leichten Fehlern:** Für diese Fehler werden 200 Treibstoffpunkte abgezogen (Zeile 1830). Nach der Anzeige muß man mit den nächsten drei Schritten den Fehler beheben, sonst wird der Fehler noch einmal geahndet (Zeile 1820).

Aus Erfahrung weiß ich, daß ein Anfänger große Schwierigkeiten mit dem Flugzeug haben wird. Das geht oft so weit, daß er aufgeben möchte.

Ich kann aber versichern, daß alle Werte, zum Beispiel für die Länge der Startbahn, ausreichen, um das Flugzeug sicher zu starten, zu fliegen und zu landen. Dazu gehört allerdings Konzentration und eine Portion Erfahrung.

## Tips von einem »Profi«

Handeln Sie überlegt und ruhig, denn wer unruhig wird und hastig handelt, ist schon so gut wie abgestürzt. Behalten Sie die Anzeigetafel immer im Auge und gehen Sie zur Landung rechtzeitig mit Geschwindigkeit und Flughöhe herunter. Leichte Fehler kosten Treibstoff und eine Landung ohne Treibstoff geht mit 30prozentiger Sicherheit schief. Beachten Sie also möglichst alle Flugregeln. Auch wenn das Spiel so manchen zum Wahnsinn treiben wird (ein paar meiner Freunde hat es schon erwischt), möchte ich allen angehenden »Computerpiloten« viel Spaß und einen guten Flug wünschen.

(Michael Lyhs/wb)

| | |
|---|---|
| FL | Entfernung zum Flughafen |
| FLEI | Flugstrecke nach dem Startvorgang |
| A$() | Texttafeln |
| K | Rückgabevariable im »CALL KEY«-Befehl |
| S | Statusvariable im »CALL KEY«-Befehl |
| SCHW | Schwierigkeitsgrad (1 oder 2) |
| FLDU | Entfernung zum übernächsten Flughafen |
| B | Geschwindigkeit |
| ST | Start- beziehungsweise Landebahnlänge |
| TB | Treibstoffvorrat |
| Y1 | Position des Flugzeugsprite (Spalte) |
| J | Textvariable |
| STS | Startvorgang (=0) |
| PPP | Landemanöver (=1) |
| LRAD | Position des Landungsradarpunktes |
| FLN | Entfernung zum Notlandeplatz |
| NO | Länge der Notlandebahn |
| ZU | Stellung des Flugzeugs |
| VO | Sinkgeschwindigkeit bei Notlandung |
| RAD | Position des Radarpunktes |
| RRR | Höhe des Flugobjekts auf Gegenkurs |
| BEH | Höhe der Berge |
| FALL | Position der Landefläche für Schleudersitz |
| HINN | Indikator für Nebel (=1) |
| HIN | Hindernisse |
| BEK | Indikator für Berge (=1) |

**Variablenliste**

| | |
|---|---|
| #1,28 | Wolken |
| #4,5,7 | Häuser beziehungsweise Bäume |
| #2 | Flugzeug |
| #3 | Seitenstreifen, Schleudersitz oder Flugobjekt |
| #8 | Seitenstreifen |
| #21 | Radarpunkt |
| #10 | explodierendes Flugzeug |

**Sprites und deren Bedeutung**

**Listing zum »Flugsimulator«**

```
1 ! FLUGSIMULATOR
2 !       VON
3 ! MICHAEL LYHS
4 ! KARLSTR. 35
5 ! 6350 BAD NAUHEIM
6 ! TEL.:06032/6539
7 !   *APRIL 1984*
100 ! FARBEN
110 CALL COLOR(11,15,1,3,3,1,4,3,1,5,3,1
,6,3,1,7,3,1,8,3,1,9,3,1):: CALL SCREEN(
2)
120 ! TITELBILD
130 DISPLAY AT(1,5)BEEP ERASE ALL:">> FL
UGSIMULATOR<<";;;;;;;;;;"VON";;;;;"MICHA
EL LYHS"
140 DISPLAY AT(14,1):"WENN DU MIT DEM FL
UGZEUG    SCHON VERTRAUT BIST, DANN   DR
UECKE >>1<<"
150 DISPLAY AT(18,1):"WENN DU ANFAENGER
BIST,     DANN DRUECKE >>2<< " :: DISPLA
Y AT(24,7)BEEP:"BITTE WARTEN"
160 CALL MAGNIFY(4):: RANDOMIZE :: FL=IN
T(90000*RND)+9999 :: FLEI=FL-3000
170 ! MASKE
180 DATA 40,FFFFFFFFFFFFFFFF,119,FFFFFFF
FFFFFFFFF,120,FFFFFFFFFFFFFFFF,124,FF,12
5,0,126,0,127,0,118,FFFF,92,C0C
190 DATA 100,0,101,70F0B6671A1008C80,103
,00307868702D3322,96,FFFFFFFFFFFFFFFF,93
,0,94,0,95,0
200 DATA 104,024822141945221989671F00000
0000000004080428804089066283OE0,36,01000
18FFFC300030
210 DATA 108,000000000003020302032273FAA
```

```
BFAEB000F1F1F1DFFADFFADFFADFFDDFFCFCF,38
,FF10FCF2FAFC88FC0
220 DATA 128,00003C5FFF0100000,130,00010
3FFFFF0000000,140,00385EFF1F21,141,0,142
,000000E1FBFF47A00
230 DATA 132,00003C5FFF2100200,134,00010
3FFFFF040A00,136,00030F7FFFFF7F3F,138,00
80FCFEFFFFFEFC0,113,385C3C,114,0,115,0,1
12,0
240 DIM A$(40):: FOR I=1 TO 33 :: READ V
,B$ :: CALL CHAR(V,B$):: NEXT I :: RESTO
RE 1630 :: FOR I=1 TO 18 :: READ A$(I)::
 NEXT I
250 ! SCHWIERIGKEITSGRAD
260 CALL HCHAR(24,1,32,32):: CALL KEY(0,
K,S):: IF K=50 THEN SCHW=1 :: FL=10000 :
: FLEI=7000 :: GOTO 270 ELSE IF K=49 THE
N 270 ELSE 260
270 FOR I=1 TO 10 :: CALL HCHAR(I,1,120,
32):: NEXT I
280 FOR I=11 TO 16 :: CALL HCHAR(I,1,40,
32):: NEXT I
290 GOTO 320
300 IF FLDU THEN FL=FLDU :: FLEI=FLDU-20
00 ELSE FL=20000 :: FLEI=17000
310 ! ANFANGSBILD FLUGHAFEN
320 FOR I=18 TO 24 :: CALL HCHAR(I,3,32,
28):: NEXT I :: CALL COLOR(12,5,1,2,3,1)
330 CALL SPRITE(#8,124,7,129,1,#3,124,7,
129,130,#2,132,2,113,153,#4,108,15,60,30
,#5,108,15,55,90)
340 CALL HCHAR(17,1,118,32)
350 CALL SPRITE(#7,108,15,70,200,#1,136,
16,10,1,0,B-8,#28,136,16,5,130,0,B-8)
360 CALL DELSPRITE(#10,#21)
370 STS,B=0 :: ST=3000 :: TB=4000
380 GOSUB 1860
390 ! STARTEN UND FLUG
400 AFALL,PPP,ASD,HIN=0
410 CALL KEY(0,K,S):: CALL POSITION(#2,Y
1,X1):: CALL SOUND(-4000,B+130,15,-6,24)
:: CALL MOTION(#2,0,0)
420 DISPLAY AT(19,22):(Y1-113)*-10 :: TB
=TB-3 :: DISPLAY AT(22,22):TB
430 IF BERG=0 OR ASD=0 THEN 440 ELSE IF
FL<=ENT+300 AND Y1>113-BEH/10 THEN J=8 :
: GOTO 770 ELSE IF FL<ENT THEN BERG,BEK=
0 :: HIN=10
440 IF STS THEN 450 ELSE ST=ST-B*3 :: DI
SPLAY AT(20,22):ST :: IF Y1<113 THEN GOT
O 1910 ELSE IF ST<=0 THEN J=1 :: GOTO 16
10 ELSE GOTO 470
450 IF TB<=0 THEN GOTO 1130 ELSE IF FL<2
000 AND PPP=0 THEN GOSUB 1950
460 FL=FL-B*2 :: DISPLAY AT(21,22):FL ::
 IF FL<=0 THEN 630 ELSE IF HIN=10 AND RN
D<.05 THEN 1350 ELSE IF Y1=113 THEN J=11
 :: GOTO 770
470 IF S=0 THEN 410 ELSE IF STS=0 THEN 5
20
480 FEL=FEL+1 :: IF Y1>103 AND ZU=3 THEN
 J=6 :: GOSUB 1820 ELSE IF B<40 THEN J=4
 :: GOSUB 1820 ELSE IF Y1<100 THEN HIN=1
0
490 IF ZU<>3 AND Y1<93 THEN J=5 :: GOSUB
 1820 ELSE IF FL<FLEI THEN ASD=1 ELSE IF
 ZU<>3 AND B>50 THEN J=5 :: GOSUB 1820
500 IF Y1<2 OR Y1>113 THEN J=12 :: GOTO
770
510 IF Y1>55 AND B>60 THEN J=3 :: GOSUB
1820 ELSE IF Y1<43 AND B<80 THEN J=4 ::
GOSUB 1820
520 IF B<38 THEN 570
530 IF K=49 THEN CALL PATTERN(#2,132)::
ZU=1 :: GOTO 410 ELSE IF K=51 THEN CALL
PATTERN(#2,128):: ZU=3 :: GOTO 410
540 IF K=50 THEN CALL PATTERN(#2,140)::
ZU=2 :: IF FL>1000 AND STS=1 THEN J=10 :
: GOSUB 1820 ELSE GOTO 410
550 IF STS=0 AND ZU<>2 THEN 570
560 IF K=88 THEN CALL MOTION(#2,3,0):: G
OTO 410 ELSE IF K=69 THEN CALL MOTION(#2
,-3,0):: TB=TB-5 :: GOTO 410
570 IF K=90 THEN B=B+1 :: TB=TB-3 :: GOT
O 580 ELSE IF K=46 THEN B=B-1
580 IF B>120 THEN B=B-2
590 CALL MOTION(#4,0,B,#5,0,B,#7,0,B,#3,
0,B,#8,0,B,#1,0,B-8,#28,0,B-8):: DISPLAY
 AT(18,22):B*5
600 IF K=48 AND STS=0 THEN 1220
610 IF PPP=0 THEN 410 ELSE IF K=83 THEN
LRAD=LRAD-2 ELSE IF K=68 THEN LRAD=LRAD+
2 ELSE GOTO 410
620 CALL SOUND(-50,400,6):: CALL LOCATE(
#21,164,LRAD):: GOTO 410
630 ! LANDUNG
640 CALL SPRITE(#8,124,9,129,1,0,B,#3,12
4,9,129,130,0,B):: CALL HCHAR(21,12,32,2
0):: CALL COLOR(12,5,1,2,3,1)
650 CALL COLOR(#4,15,#5,15,#7,15,#2,2)
660 CALL PATTERN(#4,108,#5,108,#6,108,#7
,108):: FLDU=INT(RND*40000)+10000 :: DIS
PLAY AT(24,1)BEEP:"NAECHSTER FLUGHAFEN:
";FLDU
670 CALL KEY(0,K,S):: CALL MOTION(#2,0,0
):: CALL POSITION(#2,Y1,X1):: CALL SOUND
(-4000,B+130,15,-6,24):: DISPLAY AT(20,2
2):ST
680 DISPLAY AT(19,22):(Y1-113)*-10 :: ST
=ST-B*2 :: IF ST<=0 AND Y1<>113 THEN J=1
 :: GOTO 1610 ELSE IF B<=0 THEN J=18 ::
GOTO 1170
690 TB=TB-3 :: DISPLAY AT(22,22):TB :: I
F S=0 THEN 670
700 IF K=88 THEN CALL MOTION(#2,2,0):: G
OTO 670 ELSE IF K=69 THEN CALL MOTION(#2
,-8,0):: GOTO 670
710 IF Y1>113 THEN J=13 :: GOTO 770 ELSE
 IF Y1<113 THEN 720 ELSE IF LRAD<48 OR L
RAD>54 THEN J=9 :: GOTO 770
720 IF B<30 AND ZU<>1 THEN J=14 :: GOTO
770 ELSE IF ZU=2 THEN TB=TB-10
730 IF K=49 THEN CALL PATTERN(#2,132)::
ZU=1 :: GOTO 670 ELSE IF K=46 THEN B=B-1
740 IF K<>90 THEN 750 ELSE FL=FLDU :: FL
EI=FLDU-3000 :: CALL SOUND(-3000,-6,1)::
 GOTO 1910
750 CALL MOTION(#8,0,B,#3,0,B,#4,0,B,#5,
0,B,#7,0,B):: DISPLAY AT(18,22):B*5
760 IF K=48 THEN 1220 ELSE 670
```

**Listing zum »Flugsimulator« (Fortsetzung)**

```
770 ! MELDUNG VON SCHWEREN FEHLERN
780 CALL DELSPRITE(#21):: FOR I=18 TO 24
 :: CALL HCHAR(I,1,32,32):: NEXT I :: DI
SPLAY AT(19,3):A$(J)
790 CALL MOTION(#2,0,0,#8,0,0,#3,0,0,#4,
0,0,#5,0,0,#7,0,0,#25,0,0):: CALL SOUND(
-1000,-7,5):: B=0
800 CALL SPRITE(#10,104,2,Y1-12,X1):: CA
LL SOUND(1000,-6,5)
810 CALL DELSPRITE(#2):: FOR I=1 TO 1100
 :: NEXT I :: GOTO 300
820 ! NOTLANDUNG
830 FLN=(Y1-113)*-65+B*3 :: NO=2100 :: D
ISPLAY AT(20,10):"NOTLANDEBAHN:";NO :: D
ISPLAY AT(21,9):"NOTLANDEPLATZ:";FLN
840 CALL KEY(0,K,S):: CALL MOTION(#2,1,0
):: CALL POSITION(#2,Y1,X1):: CALL SOUND
(-1000,B+130,7,-6,20)
850 CALL MOTION(#4,0,B,#5,0,B,#7,0,B)
860 IF B<=20 THEN 870 ELSE IF ZU=2 THEN
B=B-2 ELSE B=B-1
870 DISPLAY AT(19,22):(Y1-113)*-10 :: DI
SPLAY AT(18,22):B*5 :: IF B<=0 AND Y1<>1
13 THEN J=9 :: GOTO 770
880 FLN=FLN-B*3 :: DISPLAY AT(21,23):FLN
 :: IF FLN<=0 THEN 920 ELSE IF Y1>113 TH
EN J=13 :: GOTO 770 ELSE IF S=0 THEN 840
890 IF K=50 THEN CALL PATTERN(#2,140)::
ZU=2 :: GOTO 840 ELSE IF K=88 THEN CALL
MOTION(#2,20,0):: GOTO 840
900 IF K=51 THEN CALL PATTERN(#2,128)::
ZU=3 :: GOTO 840 ELSE IF K=69 THEN CALL
MOTION(#2,-20,0):: GOTO 840 ELSE IF K=46
 THEN B=B-1
910 IF K=48 THEN 1220 ELSE 840
920 VO=1 :: DISPLAY AT(21,1):" "
930 CALL KEY(0,K,S):: CALL MOTION(#2,VO,
0):: CALL SOUND(-1000,B+130,5,-6,15):: N
O=NO-B :: DISPLAY AT(20,22):NO :: CALL P
OSITION(#2,Y1,X1)
940 DISPLAY AT(19,22):(Y1-113)*-10
950 IF Y1>113 THEN J=13 :: GOTO 770 ELSE
 IF NO<=0 THEN J=1 :: GOTO 770 ELSE IF Y
1>105 AND ZU=3 THEN J=6 :: GOSUB 1830
960 IF Y1>=105 THEN VO=0
970 IF S=0 THEN 930
980 IF K=88 THEN CALL MOTION(#2,5,0):: G
OTO 930
990 IF K=49 THEN CALL PATTERN(#2,132)::
ZU=1 :: GOTO 930 ELSE IF K=50 THEN CALL
PATTERN(#2,140):: ZU=2 :: GOTO 930 ELSE
IF K=46 THEN B=B-1
1000 CALL MOTION(#4,0,B,#5,0,B,#7,0,B)
1010 DISPLAY AT(18,22):B*5 :: GOTO 930
1020 ! FLUGOBJEKT AUF GEGENKURS
1030 CALL KEY(0,K,S):: CALL POSITION(#2,
Y1,X1):: CALL SOUND(-3000,ABS(B)+130,15,
-6,24):: DISPLAY AT(19,22):(Y1-113)*-10
:: TB=INT(TB-Y1/15)
1040 CALL MOTION(#2,0,0):: FL=FL-B*2 ::
DISPLAY AT(21,22):FL :: IF FL<RTR-4000 T
HEN 1550 ELSE DISPLAY AT(22,22):TB
1050 RRR=164+(RAD-Y1)/2 :: CALL LOCATE(#
21,RRR,51):: IF RND<.1 THEN RAD=RAD+2 EL
SE IF RND<.2 THEN RAD=RAD-2
1060 IF S=0 THEN 1030
1070 IF B<40 THEN J=4 :: GOSUB 1820 ELSE
 IF Y1<2 THEN J=12 :: GOTO 770 ELSE IF Y
1>55 AND B>60 THEN J=3 :: GOSUB 1820
1080 IF Y1<43 AND B<80 THEN J=3 :: GOSUB
 1820 ELSE IF FL<=0 THEN 630
1090 IF K=88 THEN CALL MOTION(#2,3,0)::
GOTO 1030 ELSE IF K=69 THEN CALL MOTION(
#2,-3,0):: TB=TB-10 :: GOTO 1030 ELSE IF
 K=90 THEN B=B+1
1100 IF K=46 THEN B=B-1
1110 IF B>125 THEN B=B-2
1120 CALL MOTION(#4,0,B,#5,0,B,#7,0,B)::
 DISPLAY AT(18,22):B*5 :: GOTO 1030
1130 ! TREIBSTOFF VERBRAUCHT
1140 IF FL<=6000 THEN 1720
1150 FOR I=1 TO 15 :: DISPLAY AT(23,3):A
$(15):: CALL SOUND(-100,500,7):: FOR IU=
1 TO 15 :: NEXT IU :: DISPLAY AT(23,1):"
 " :: NEXT I
1160 GOTO 830
1170 CALL COLOR(#2,2,#8,12,#3,12):: CALL
 COLOR(12,5,1,2,13,1):: FOR I=10 TO 300
STEP 10 :: CALL SOUND(-1000,110+I,10,250
+I,5):: NEXT I
1180 FOR I=18 TO 24 :: CALL HCHAR(I,1,32
,32):: NEXT I
1190 DISPLAY AT(18,1):A$(J):: FOR I=1 TO
 400 STEP 10 :: CALL SOUND(-4000,200+I,1
,400+I,1):: NEXT I
1200 GOTO 300
1210 !SCHLEUDERSITZ
1220 CALL SOUND(-1000,-6,0):: CALL SPRIT
E(#4,112,2,Y1-10,150,-15,0):: FALL=INT(R
ND*9)+13 :: CALL HCHAR(17,FALL,32,3)
1230 CALL MOTION(#2,1,-B,#3,0,0,#5,0,0,#
8,0,0,#7,0,0)
1240 CALL SPRITE(#10,104,2,Y1,10):: CALL
 DELSPRITE(#2)
1250 FOR I=-10 TO 7 :: CALL MOTION(#4,I,
-1):: CALL SOUND(-100,-5,ABS(I)):: NEXT
I :: CALL DELSPRITE(#10)
1260 CALL KEY(0,K,S):: CALL MOTION(#4,I,
0):: CALL SOUND(-4250,-5,20):: CALL POSI
TION(#4,Y1,X1):: IF Y1>106 AND Y1<120 TH
EN 1310
1270 IF S=0 THEN 1260
1280 IF K=70 THEN CALL CHAR(112,"3C5AF7F
FA5522C1"):: CALL SOUND(-200,-4,0):: AFA
LL,I=1
1290 IF K=83 THEN CALL MOTION(#4,1,-2)::
 GOTO 1260 ELSE IF K=68 THEN CALL MOTION
(#4,1,2)
1300 GOTO 1260
1310 CALL MOTION(#4,0,0)
1320 IF X1<FALL*8-8 OR X1>(FALL+3)*8-8 T
HEN J=16 :: GOTO 770 ELSE IF AFALL=0 THE
N J=16 :: GOTO 770
1330 J=17 :: CALL CHAR(112,"0"):: GOTO 1
170
1340 ! ERZEUGUNG DER HINDERNISSE
1350 IF SCHW=1 OR BEK=1 THEN 410
1360 ON INT(RND*11)+1 GOTO 1480,1440,152
0,1410,1520,1480,1440,1440,1480,1440,137
0
```

```
1370 IF HINN=1 THEN 410
1380 HINN=1 :: HIN=0 :: FOR I=1 TO 10 ::
 DISPLAY AT(23,1):"WARNUNG ! NEBEL KOMMT
 AUF" :: CALL SOUND(-100,300,3):: FOR IU
=1 TO 20 :: NEXT IU
1390 CALL HCHAR(23,1,32,32)
1400 NEXT I :: CALL COLOR(#2,15,#4,15,#5
,15,#7,15):: CALL COLOR(2,15,1,12,15,1):
: GOTO 410
1410 IF RND<.9 THEN 410
1420 HIN=2 :: FOR I=1 TO 15 :: DISPLAY A
T(23,1):"  TRIEBWERK AUSGEFALLEN !    NO
TLANDEN !"
1430 CALL SOUND(-100,400,3):: FOR IU=1 T
O 20 :: NEXT IU :: DISPLAY AT(23,1):" "
:: NEXT I :: GOTO 830
1440 IF HINN<>1 OR FL<20000 THEN 410
1450 BEH=INT(RND*601)+400
1460 BEK=1 :: HIN=0 :: FOR I=1 TO 25 ::
DISPLAY AT(23,1):"BERG MIT EINER HOEHE V
ON";BEH;"IN 5000M ENTFERNUNG !" :: CALL
SOUND(-100,400,6)
1470 NEXT I :: BERG=1 :: ENT=FL-5000 ::
DISPLAY AT(23,1):" ":" " :: GOTO 410
1480 IF FL<7000 THEN 410
1490 FOR I=1 TO 10 :: DISPLAY AT(23,3):"
FLUGOBJEKT AUF GEGENKURS !" :: CALL SOUN
D(-100,360,9)
1500 FOR IU=1 TO 15 :: NEXT IU :: DISPLA
Y AT(23,1):" " :: NEXT I :: GOSUB 1890 :
: CALL SPRITE(#21,92,12,164,51):: CALL H
CHAR(21,7,45)
1510 RAD=Y1 :: RTR=FL :: GOTO 1030
1520 IF HINN<>1 THEN 410
1530 CALL SOUND(-400,300,14):: CALL COLO
R(#2,2,#4,4,#5,4,#7,4):: CALL COLOR(12,5
,1,2,9,1):: HINN=0 :: GOTO 410
1540 ! ERZEUGUNG DES FLUGOBJEKTS
1550 HOH=Y1-(164-RRR)*2 :: CALL SPRITE(#
3,36,15,HOH,1,0,B+5)
1560 IF HOH<Y1-6 OR HOH>Y1+6 THEN GOTO 1
570 ELSE GOTO 1600
1570 CALL COINC(#3,RAD,242,4,GEG):: IF G
EG<>-1 THEN 1570
1580 CALL SOUND(-4000,ABS(B)+130,15,-6,2
4)
1590 CALL DELSPRITE(#3,#21):: FOR I=18 T
O 24 :: CALL HCHAR(I,4,32,7):: NEXT I ::
 GOTO 410
1600 FOR I=1 TO 122-B :: NEXT I :: J=7 :
: GOTO 770
1610 CALL SPRITE(#3,108,2,95,30,0,B):: G
OTO 780
1620 ! TEXTTAFELN
1630 DATA STARTBAHN UEBERSCHRITTEN,FLUGB
EREICH WURDE VERLASSEN,GESCHWINDIGKEIT Z
U HOCH,GESCHWINDIGKEIT ZU GERING
1640 DATA FAHRWERK WURDE NICHT
EINGEZOGEN,FAHRWERK MUSS IN DIESER HOEHE
     AUSGEFAHREN SEIN,FLUGZEUG BEI ZUSAM
MENSTOSS  EXPLODIERT
1650 DATA FLUGZEUG HAT BERG GERAMMT UND
IST EXPLODIERT,DURCH FEHLVERHALTEN NOTLA
NDUNG GESCHEITERT
1660 DATA FAHRWERK DARF NOCH NICHT AUSGE
FAHREN WERDEN
1670 DATA FLUGZEUG HAT VOR DER
 LANDEBAHN AUFGESETZT
1680 DATA FLUGBEREICH WURDE VERLASSEN,FL
UGZEUG HAT ZU HART          AUFGESETZT,FLU
GZEUG DARF BEIM AUSROLLEN NICHT SCHRAEG
STEHEN
1690 DATA TREIBSTOFF VERBRAUCHT
 NOTLANDEN,LANDUNG MIT DEM FALLSCHIRM
FEHLGESCHLAGEN
1700 DATA DURCH DEN ABSPRUNG HAST DU   D
EIN LEBEN GERETTET. DEIN   FLUGZEUG IST
EXPLODIERT
1710 DATA BRAVO! DU HAST DAS FLUGZEUG
SICHER GELANDET
1720 ! LANDUNG OHNE                  TRTR
STOFF
1730 VO=1
1740 CALL KEY(0,K,S):: CALL MOTION(#2,VO
,0):: CALL SOUND(-1000,B+130,15,-6,24)::
 FL=FL-B*2 :: DISPLAY AT(21,22):FL
1750 DISPLAY AT(19,22):(Y1-113)*-10 :: I
F FL<=0 THEN 630 ELSE IF Y1=113 THEN J=1
1 :: GOTO 770
1760 IF S=0 THEN 1740
1770 IF K=50 THEN CALL PATTERN(#2,140)::
 ZU=2 :: GOTO 1740 ELSE IF K=49 THEN CAL
L PATTERN(#2,132):: ZU=1 :: GOTO 1740
1780 IF K=69 THEN CALL MOTION(#2,-20,0):
: GOTO 1740
1790 IF K=46 THEN B=B-1 :: CALL MOTION(#
4,0,B,#5,0,B,#7,0,B)
1800 IF K=48 THEN 1220 ELSE 1740
1810 ! MELDUNG LEICHTER FEHLER
1820 IF FEL<3 THEN RETURN
1830 CALL SOUND(-700,333,5,-4,8):: CALL
DELSPRITE(#21):: DISPLAY AT(23,1):A$(J):
: TB=TB-200
1840 FEL=0 :: FOR I=1 TO 80 :: NEXT I ::
 FOR I=23 TO 24 :: CALL HCHAR(I,1,32,32)
:: NEXT I :: RETURN
1850 ! ANZEIGETAFEL
1860 DISPLAY AT(18,16):"SPEED:";B*5;::TA
B(12);"FLUGHOEHE:";::TAB(12);"STARTBAHN:
";ST;::TAB(12);"FLUGHAFEN:";FL;::TAB(11)
;"TREIBSTOFF:";TB
1870 CALL HCHAR(23,1,32,2):: RETURN
1880 ! RADARSCHIRM
1890 FOR I=19 TO 23 :: CALL HCHAR(I,4,96
,7):: NEXT I :: FOR I=4 TO 10 STEP 6 ::
CALL VCHAR(19,I,42,5):: NEXT I
1900 FOR I=18 TO 24 STEP 6 :: CALL HCHAR
(I,4,42,7):: NEXT I :: RETURN
1910 ! FLUGLANDSCHAFT
1920 ST=3000 :: CALL PATTERN(#4,100,#5,1
00,#7,100):: DISPLAY AT(20,1):" " :: CAL
L DELSPRITE(#8,#3,#1,#28):: GOSUB 1860 :
: STS=1
1930 CALL COLOR(2,7,1,12,5,1):: CALL COL
OR(#2,2,#4,4,#5,4,#7,4):: DISPLAY AT(24,
1):" " :: GOTO 400
1940 ! LANDUNGSRADAR
1950 GOSUB 1890 :: LRAD=INT(RND*31)+35 :
: PPP=1 :: CALL VCHAR(19,7,73,5):: CALL
SPRITE(#21,92,12,164,LRAD):: RETURN
```

**Listing zum »Flugsimulator« (Schluß)**

# Klangroutinen

## Wie man einen TI99/4A als Sirene mißbraucht und schließlich Tonfrequenzen exakt berechnet, schildert dieser Bericht.

Die einfachste Veränderung eines Klangs erreicht man, indem man die Lautstärke eines Tons gezielt manipuliert. Wenn man einen Ton ständig an- und abschwellen läßt, entsteht ein sirenenartiger Effekt. Nicht minder wirkungsvoll verändert man Klänge wenn man mit einer Schleife die Lautstärke des Tons langsam reduziert. Dadurch kommt ein Ton zustande, der sich wie ein Gong anhört. Das erste Listing zeigt ein Beispiel. Durch Ändern der Variablen »F« in Zeile 110 erreicht man verschiedene Tonhöhen.

```
100 REM Gong-Routine
110 F=880
120 FOR N=0 TO 20 STEP 2
130 CALL SOUND(-100,F,N)
140 NEXT N
150 END
```

Diese Routine kann in ein kleines Programm eingebaut werden. Die Frequenzen werden jetzt aus DATA-Zeilen gelesen. Die Frequenz 30000, die dort auftaucht, dient zur Erzeugung einer Pause. Auf diese Weise muß man die Schleife bei kurzen Pausen nicht unterbrechen und kann auch die Pausendauer genau dosieren.

```
100 REM BIG BEN
110 FOR N=1 TO 9
120 READ A,B
130 FOR L=0 TO 20 STEP 2
140 CALL SOUND(-B,A,L)
150 NEXT L
160 NEXT N
170 STOP
180 DATA 195,100,155,100
190 DATA 174,100,116,400
200 DATA 30000,100,116,100
210 DATA 174,100,195,100
220 DATA 155,100
```

Um längere Melodien wiederzugeben, ist das ständige Wiederholen des »CALL SOUND«-Befehls sicher eine undankbare Lösung. Als Alternative bietet sich natürlich das Lesen aus DATA-Zeilen an. Doch hier gibt es mehrere Methoden des Einlesens. Im Listing »Big Ben« werden die Frequenzen mit den zugehörigen Tonlängen aus den »DATA«-Zeilen gelesen und direkt im »CALL SOUND«-Befehl verarbeitet. Dies funktioniert allerdings nur ab einer gewissen Tonlänge. Bei sehr kurzen Tönen ist die Verarbeitungsgeschwindigkeit des TI99/4A bei der Abarbeitung des »READ«-Befehls zu gering. Es entstehen daher ungewollte Pausen zwischen den einzelnen Tönen. Diesen Nachteil gleicht man aus, indem man die Daten zuvor in einem Feld speichert. Auf das vorige Listing angewandt, sähe das dann so aus:

```
100 REM BIG BEN 2
110 DIM T(9,1)
120 FOR N=1 TO 9
130 READ F,L
140 T(N,0)=F
150 T(N,1)=L
160 NEXT N
170 FOR N=1 TO 9
180 FOR A=0 TO 20 STEP 2
190 CALL SOUND(-T(N,1),T(N,0),A)
200 NEXT A
210 NEXT N
220 STOP
230 DATA ...
```

Die dritte und sehr elegante Methode berechnet die benötigten Frequenzen. Hierfür gibt es eine einfache Formel, die, ausgehend von einem Grundton, eine komplette Tonleiter erzeugt.

```
100 REM TONLEITER
110 F=440
120 FOR N=0 TO 12
130 T=F*(2^(1/12))^N
140 CALL SOUND(-100,T,2)
150 NEXT N
160 END
```

Speichert man nun die so berechneten Frequenzen in einem Feld, liegt der Vorteil klar auf der Hand. Man muß nicht mehr jede einzelne Frequenz in die DATA-Zeilen eintippen, sondern kann sich auf die Feldindizes für die Tonfolge der Melodie beschränken. Auch die Tonlängen lassen sich auf diese Weise abfragen. Gerade bei längeren Musikstücken stellt dies eine wesentliche Zeit- und Platzersparnis dar.

```
100 REM FREQUENZBERECHNUNG
110 DIM T(12),L(4)
120 FOR N=1 TO 4
130 L(N)=N*100
140 NEXT N
150 FOR N=0 TO 12
160 T(N)=440*(2^(1/12))^N
170 NEXT N
180 FOR N=0 TO 12
190 READ A,B
200 CALL SOUND(-L(B),T(A),2)
210 NEXT N
220 STOP
230 DATA 0,1,1,1,2,1,3,1,4,1,5,1
240 DATA 6,1,7,1,8,1,9,1,10,1
250 DATA 11,1,12,4
```

(Wolfgang Czerny/wb)

# BÜCHER

## Musisches und praktisches für den TI 99/4A

Hinter dem schlichten Titel »Programme für den TI 99/4A« verbirgt sich eine reine Programm-Sammlung. Auch einige nützliche Tips über Sprites und über Sortieralgorithmen findet man in diesem Buch. Die Programme sind sowohl in Basic als auch in Extended Basic geschrieben. Die Themenwahl ist bunt gemischt und reicht von Geschäftsprogrammen über die Mathematik bis hin zu den unvermeidbaren Spielen.

Der Klangerzeugung ist das Buch »Musik mit dem TI 99/4A« gewidmet. Im Grunde genommen behandelt dieses Buch nur vier Programme. Diese werden jedoch ausführlich beschrieben. Zahlreiche Flußdiagramme, Variablenlisten und einführende Theorie gehen jedem der Programme voraus. Behandelt werden die Klangsynthese, sowie die Ein- und Ausgabe von Melodien. Alle Programme sind in Basic gehalten, so daß sie für jeden TI-Besitzer verwendbar sind.

Das Buch »Textverarbeitung mit Programmen für TI 99/4A und VC 20« versteht sich als Grundlagenlektüre zum Thema eigene Textverarbeitung. Ausführlich wird die Stringmanipulation und die Verwendung der Drucksteuerzeichen erläutert. Zahlreiche kurze Beispielprogramme zu jedem Unterpunkt veranschaulichen, wie das Gelernte praktisch eingesetzt wird. Dabei wird immer auf die Unterschiede zwischen den beiden verwendeten Computern eingegangen. Den Abschluß bildet ein fertiges Textverarbeitungsprogramm für den TI 99/4A.

(Wolfgang Czerny/wb)

R.Heigenmoser, »Programme für den TI 99/4A«, Hofacker Verlag, ISBN 3-88963-149-8, 49 Mark
Eugen Gehrer, »Musik mit dem TI-99/4A«, Vieweg Verlag, ISBN 3-528-04277-X, 48 Mark
Arnim und Ingeborg Tölke, »Textverarbeitung«, Vieweg Verlag, ISBN 3-528-04276-1, 36 Mark

## Rundum informativ

Der Einstieg in die Welt des Computers ist schwer genug. Peter Rodwell gehört mit seinem Buch »Home Computer klipp und klar« zu den Autoren, die wirklich umfassend und verständlich zu informieren verstehen. Vielleicht liegt es an seinem fundierten Wissen. Wenn dann ein Verlag noch eine durchgehend gute Illustration spendiert ist ein empfehlenswertes Buch entstanden! Zwar wäre mancher Schnörkel besser unterblieben; Hauptsache ist aber, daß die Grafiken und Bilder den Konsum des doch sehr anstrengenden Stoffs erleichtern. Der knapp gehaltene Text tut sein übriges und läßt kein Thema aus, egal ob Abstraktes wie Hexadezimal-Arithmetik und Logik, Grundlegendes wie Programmieren in Basic, interessantes Nebensächliches wie Grafiktablett oder Praktisches wie Textverarbeitung. Alles wird beschrieben. Über 200 spannende Seiten, die umfassend informieren, für 29,80 Mark — das ist wirklich ein Lob wert.

(lg)

Peter Rodwell, »Home Computer klipp und klar«, Otto Maier Verlag, Ravensburg, ISBN 3-473-42625-3, Preis 29,80 Mark

## Hardware-Basteln leicht gemacht

Auch ein Computerfreak kann manchmal in die Lage kommen, zum Schraubenzieher und Lötkolben greifen zu müssen. Einer der bekanntesten Autoren für Elektronik-Basteleien ist Dieter Nührmann, geschätzt wegen seines lockeren Tons, auch in trockenen Fragen und seinen Tips aus der Trickkiste. Besonders die Stoffauswahl seiner Bücher trifft den Bedarf der Hobby-Elektroniker immer sehr genau. Auch »das kleine Werkbuch Elektronik« ist eine echte Fundgrube. Entstanden als »Hobby«-Version des »großen Werkbuchs«, das für professionelle Anwendung eine Art Standardwerk ist, enthält der neue Band praktisch alle Daten, Formeln, Grundschaltungen und Praxistips, die sich ein Bastler wünschen kann. Etwas knapp geraten ist lediglich das Sachwortverzeichnis. Für den normalen Hobby-Bastler ist das 435 seitige Buch ein handliches und umfassendes Nachschlagewerk, das allerdings mit 48 Mark etwas teuer ist.

(lg)

Ing. Dieter Nührmann, »Das kleine Werkbuch Elektronik«, Franzis, ISBN 3-7723-7171-X, Preis 48 Mark

## Die ersten Bücher für die Laser-Computer

Für alle, die mit dem Handbuch der Laser-Computer nicht zufrieden sind, ist der erste Band der Reihe »Laser Home-Computer« gedacht. Von der Tastaturbelegung bis hin zu den Strukturen einfacher Basic-Befehle reicht das Spektrum. Als hilfreich besonders für Anfänger erweisen sich die Kapitel über die Befehle »FOR ... NEXT ...« und »IF ... THEN ...«.

Das Buch ist in einer gut lesbaren Sprache geschrieben. Die Themen beschränken sich jedoch auf Informationen für Anfänger. Für diese Zielgruppe ist dieser erste Band der neuen Reihe hervorragend geeignet.

Der zweite Band wendet sich an die Laser-Besitzer, die ihren Computer schon beherrschen. Viele PEEKs und POKEs sprengen den Rahmen des normalen Basic. Von Zahlen mit doppelter Genauigkeit bis zur Reaktivierung von mit »NEW« gelöschten Programmen reichen die Tips. Auch für den Freund der maschinennahen Programmierung ist dieses Buch eine Fundgrube.

Jeder der mit seinem Computer selbst programmiert, und sich dabei nicht nur auf simple Programme beschränkt, kann nur hoffen, daß die Reihe bald fortgesetzt wird. Von einem Fachmann für Laser-Computer geschrieben, stellt sie die einzige Hilfe für Besitzer der Computer von Sanyo Video dar.

(hg)

»Mein Laser Home-Computer, Tips und Tricks für Einsteiger«, Sanyo Video Vertrieb Hamburg, 24,90 Mark
Wolfgang Radeloff, »Laser Home-Computer, Software-System Handbuch I«, Sanyo Video Vertrieb Hamburg, 35 Mark

## Microdrive ohne Sprachbarriere

Jetzt gibt es auch ein Microdrive-Handbuch in deutscher Sprache. Jedoch ist das nicht der einzige positive Punkt des Werkes »Mikrodrive-Handbuch für den ZX Spectrum« von Jochen Merz. Der Autor geht auf alle Anwendungen des Interface 1, also auf das Microdrive ein, auf den Aufbau von Netzwerken, die RS232-Schnittstelle, die neue Speicherbelegung und die zusätzlichen Befehle. Die Erklärungen umfassen nicht nur das »wie« sondern erfreulicher Weise auch das »warum«.

In der zweiten Hälfte des Handbuchs wird ein erweitertes Basic in Form eines Listings, einer Beschreibung und der Maschinencode-Erklärung vorgestellt. Wem das Abtippen zu viel ist, dem kann geholfen werden: es gibt das Buch auch mit Software-Kassette (für 10 Mark mehr).

(mk)

Jochen Merz, »Mikrodrive-Handbuch für den ZX Spectrum«, Profisoft, ISBN 3-923985-04-5, 29 Mark.

# Unternehmen Galactica

**Ein Spiel, das der Fernsehserie »Raumschiff Enterprise« nachempfunden wurde. Werden Sie mit Ihrem TI 99/4 und dem in Extended Basic geschriebenen Programm zum Raumschiff-Kapitän.**

Man schreibt das Jahr 2463. Die Raumfahrttechnik hat große Fortschritte gemacht, immer mehr Siedlungen wurden im intergalaktischen Raum gegründet. Auf der Suche nach neuen Lebensräumen und Rohstoffen werden Expeditionen in ferne Galaxien entsandt. Nicht selten hatten die Menschen dabei Kontakt mit fremden, aber friedlichen Wesen.

Auf einer Forschungskolonie im Andromedanebel traf man vor drei Monaten auf Wesen, die sich Zylonen nennen. Die unscheinbaren Wesen boten seltene Metalle zum Handel an. Daraufhin wurde eine Flotte von sechs Raumschiffen zum Andromedanebel geschickt. Ihr Auftrag war es, die Metalle zu kaufen und zur Erde zu bringen. Zum Schutz vor Angriffen begleiten Sie als Commander des Raumkreuzers Galactica die Flotte auf Ihrer Mission.

Kurz nach Erreichen des Andromedanebels wurden die Raumschiffe und die Forschungskolonie von weit überlegenen Zylonverbänden angegriffen. Die Kolonie sowie vier Transportraumschiffe wurden vollkommen vernichtet.

Zusammen mit einem zweiten Schiff konnten Sie fliehen. Da das zweite Schiff stark beschädigt war, wurde es von Zylonschiffen eingeholt und nach hartem Kampf aufgegeben. Die verbliebene Besatzung und die Reste der Abfangjägergeschwader wurden von Ihnen an Bord der Galactica gebracht.

In dieser Situation befinden Sie sich, nachdem das Programm »Unternehmen Galactica« geladen und gestartet wurde.

Mitten in einer unbekannten Galaxie, verfolgt von Zylonen und mit begrenzten Vorräten an Energie. Die einzige Hilfe bei der Suche nach der Erde ist die Anzeige der Entfernung zwischen der Galactica und der Erde. Den richtigen Kurs müssen Sie bei der Ab- oder Zunahme der Entfernung herausfinden. Um Ihr Ziel zu erreichen, müssen Sie aber sicherlich mehrere Male auf unbekannten Planeten landen, um Vorräte zu ergänzen.

Besonders das Erkunden fremder Planeten macht das Spiel so spannend und abwechslungsreich. Es gibt die verschiedensten Situationen, die Sie auf den Planeten erwarten. Ist der Planet zum Beispiel bewohnt oder nicht, findet man eine höher-, gleich-, niedrigerentwickelte oder ausgestorbene Kultur vor? Trifft man funktionierende Robotersysteme an? Gibt es dort aggressive, friedliche oder handelsbereite Lebewesen?

Diese Vielfalt, kombiniert mit einer zufälligen und bei jedem Spiel anderen Kombination, bildet die Basis für das Spiel.

Um die Galactica sicher zu steuern und um in allen Situationen entsprechend zu reagieren, wird eine gewisse Übung nötig sein. Sicherlich werden Sie die Befehle nach ein paar Spielen beherrschen und Ihr Ziel erreichen.

Nach dem Starten des Programms werden zuerst die Grafikzeichen, Farben und Variablen indiziert. Danach wird der Bildschirm aufgebaut, der das Cockpit der Galactica mit verschiedenen Anzeigen simuliert. Zunächst wird aber die Sternkarte aufgebaut. Das dauert etwa drei Minuten.

Danach geht das Spiel richtig los. Der Spielablauf gliedert sich in einzelne Züge. Jeder Zug besteht aus einer Aktions- und einer Analysephase. In der Aktionsphase können die Befehle »Alarm«, »verhandeln«, nur einmal ausgeführt werden. Andere jedoch, »Sensor«, »Kurs« und »Schäden«, so oft Sie wollen.

Jeder Zug wird mit dem Befehl »NEXT« abgeschlossen und in der Analysephase untersucht. Dabei werden zum Beispiel neue Koordinaten für das Raumschiff Galactica und die Zylonen berechnet. Die Daten werden auf den aktuellen Stand gebracht.

In den Orbit eines Planeten kommt man, wenn Planet und Galactica die gleiche Position haben und die Geschwindigkeit der Galactica gleich Null ist. Planeten können nur angegriffen werden, wenn Sie sich im Orbit um den Planeten befinden. Wollen Planetenbewohner nicht verhandeln, können Sie sie durch einen Angriff zum Handeln oder zur Aufgabe zwingen. Aber Vorsicht, auch freundliche Völker sind unberechenbar.

**Die Steuerbefehle der »Galactica«**

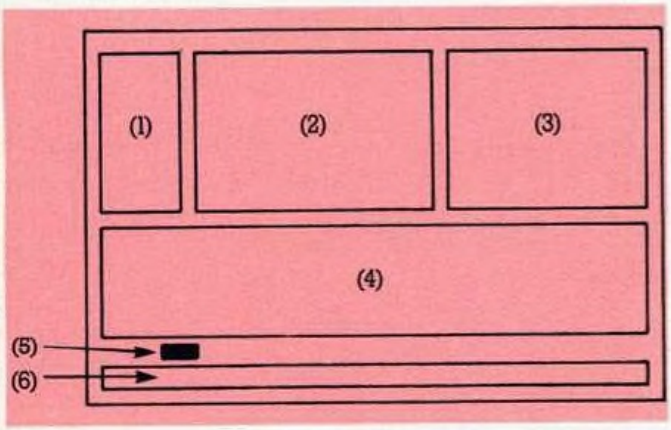
| | |
|---|---|
| **Schäden** | Schäden werden unter Energieverbrauch repariert |
| **Alarm** | Sprung ins Untermenü |
| Schutzschirm ein/aus | Schutzschirm aktivieren |
| Angreifen ja/nein | Sprung ins Untermenü |
| Jäger einsetzen ja/nein | Abfangjäger startklar machen |
| Ziel wählen | Angriffsziel wählen und |
| Zylonjäger | Jäger starten |
| Basisschiff | |
| Planet | |
| Laserkanonen ja/nein | Kanonen feuerbereit machen |
| Ziel wählen | Ziel wählen und feuern |
| Zylonjäger | |
| Basisschiff | |
| Planet | |
| **Sensor (2 Versionen)** | |
| 1) Galactica nicht im Orbit um Planet | Peilung in nähere Umgebung der Galactica. Anfliegende Raumschiffe und geortete Planeten werden gemeldet. |
| 2) Galactica im Orbit um Planet | Wie Fall 1. Zusätzliche Informationen über Name und Zivilisationsstatus des Planeten. |
| **Kurs** | Sprung ins Untermenü |
| 9 verschiedene Richtungen | 9 verschiedene Richtungen, Null bedeutet Stillstand |
| **Verhandeln** | Sprung ins Untermenü, wenn sich die Galactica im Orbit um den bewohnten Planeten befindet. |
| Handeln ja/nein | Bewohner des Planeten machen ein Angebot. Wenn Ihnen Preise angemessen erscheinen, folgt ein |
| Kaufen | Preis ok — Sprung ins Untermenü |
| Handeln | »neues Angebot« |
| Jäger kaufen | |
| Energie kaufen | |

Feilschen Sie nur um den Preis, wenn Sie noch Energie-Reserven besitzen; die Gefahr besteht, daß es zum Gefecht kommt. Der Schutzschirm benötigt viel Energie und ist nur bei Gefahr einzuschalten. Er schützt übrigens nicht vor Asteroidenfeldern. Bei Zylonenangriffen ist zu beachten, daß flüchtende Zylonjäger andere Basisschiffe verstärken. Sie haben Ihr Ziel erreicht, wenn Sie in den Orbit der Erde einschwenken.

## Der Bildschirmaufbau

Feld 1. Anzeigetafel für Zeit (jeder Zug stellt eine Zeiteinheit dar) und Entfernung zur Erde (einzige Hilfe bei der Suche nach der Erde).
Feld 2. Radarschirm in hochauflösender Farbgrafik. Es wird immer die unmittelbare Umgebung der Galactica dargestellt. Ihre Position befindet sich immer im Zentrum des Radarschirms. Die Richtung des Positionspfeils zeigt den eingeschlagenen Kurs. Zylonenschiffe sind auf dem Radarschirm als große, ovale Scheiben zu erkennen. Auch die Asteroidenfelder und Planetensysteme sind zu sehen.
Feld 3. Anzeigetafel für wichtige Daten. Einsatzbereitschaft, Geld, Anzahl der Abfangjäger und Energie werden hier angezeigt. Die Energieanzeige erfolgt analog durch ein Zeigerinstrument (e = empty / f = full).
Feld 4. Hier gibt der Bordcomputer seine Meldungen aus.
Feld 5. Die sich bewegende Anzeige zeigt den Alarmzustand der Galactica an (gelb = keine Gefahr / grün = Feinde im Anflug / rot = Angriff).
Feld 6. Hier werden die Befehle des Hauptmenüs aufgeführt.

(Gerd Herrmann/wb)




### Die umgewandelten ASCII-Codes

| Code | Funktion |
|---|---|
| 32-39 | Anzeigenumrandung |
| 40-47 | Kurspfeile |
| 128-142 | Sternkartensymbole |

### Die Verwendung der Sprites

| Sprite-Nr. | Funktion |
|---|---|
| 10 | Position der Galactica auf Radarschirm |
| 11-15 | Position der Zylonen auf Radarschirm |
| 16, 27 | Position der Erde auf Radarschirm |
| 26 | Alarmanzeige |
| 22 | Energieanzeige |

### Variablenliste

| | |
|---|---|
| a$(20) | Sternkarte |
| B$(8) | Sternname |
| c$(5) | Zivilisationsnamen |
| d$ | Leerstring |
| e$ | Lokaler Lesestring |
| g$(8) | String für Planetenzustand enthält vier Zahlenwerte |
| xe,ye | Position der Erde |
| xg,yg | Position der Galactica |
| yz(5),xz(5) | Position der Zylonengeschwader |
| sx(8),sy(8) | Position der Sterne |
| sp | Hilfsvariable für Sprungverteiler |
| p | Zeit |
| kw | Kampfwert |
| kx,ky | Kurs der Galactica |
| r | Zahl der georteten Schiffe |
| t | Zahl der georteten Sonnensysteme |
| si | Schutzschirm |
| l | Jägerzahl |
| m | Zustand |
| n | Energie |
| o | Geld |

a,b,c,d,e1,e2,e3,e4,i,j und sonstige Variablen haben nur lokale Bedeutung als Rechen- oder Laufparameter.

fa,fb,fc,fd sind logische Parameter, das heißt sie besitzen den Wert Eins oder Null. Sie sind als Wegweiser durch bestimmte Programmteile anzusehen.

### Programmbeschreibung

| | |
|---|---|
| 1000 — 1650 | Datas für Grafikzeichen und Bildschirmaufbau |
| 3000 | Lesen der Datas und Definition der Grafikzeichen |
| 4000 — 4010 | Farben setzen |
| 5000 — 5050 | Bildschirm aufbauen und Sternkarte aufstellen. Da die Sternkarte mittels einer Vektorfunktion in Abhängigkeit von ›RND‹-Werten erstellt wird, hat sie immer eine andere Form. (siehe Zeile 5022). |
| 6000 — 6050 | Startdaten werden bestimmt (Position Erde, Galactica und so weiter) und das Spiel wird gestartet. |
| 7000 — 9500 | Eigentliches Hauptprogramm! Beinhaltet Sprungverteiler (Zeile 8000 bis 8010) der die einzelnen Unterprogramme aufruft. Ebenso erfolgt die Ausgabe der aktuellen Daten (9000 bis 9500). |
| 11000 — 11029 | Unterprogramm Kurswechsel! Hier werden Kursvariablen und Richtungspfeil geändert. |
| 12000 bis 12400 | Unterprogramm Sensor! |
| 13000 — 13955 | Unterprogramm Alarm! Steuert Schutzschirm (Zeile 13000), Jägerangriff (Zeile 13020 bis 13045), Laserangriff (Zeile 13100 bis 13120), Ziel Planet (Zeile 13800 bis 13810), Ziel Basisschiffe (Zeile 13850 bis 13878), Ziel Zylonjäger (Zeile 13900), Auswertung (Zeile 13950 bis 13955). |
| 14000 — 14850 | Unterprogramm verhandeln! Steuert Kontaktaufnahme (Zeile 14000 bis 14009), Art des Kontaktes (Zeile 14010 bis 14030), handeln (Zeile 14040 bis 14075), Beibootlandung (Zeile 14800 bis 14850) |
| 15000 — 15020 | Unterprogramm Schäden! Galactica wird repariert. |

| | |
|---|---|
| 16000 — 16920 | Unterprogramm next! Hier wird die Aktionsphase analysiert. Positionswechsel der Galactica und der Zylonen, sowie Aufbau des neuen Radarschirms (Zeile 16000 bis 16100), Angriffe von Zylonen oder Planeten, Verluste, Alarmstufen und so weiter (Zeile 16200 bis 16920) |
| 17000 — 19200 | Endroutine! |
| 20001 | Error-Sprungadresse (im Programm werden Error-Meldungen zum Verzweigen oder Rücksprung zu Zeile 7000 im Zusammenhang mit dem Befehl »9 MENUE« öfters künstlich erzeugt.) |
| 30000 — 30020 | Subroutine die eine mit »a« übergebene Zeichenkette als Parameter für »CALL KEY«-Routine benutzt. |
| 30020 | Subroutine für die Ausgaben auf dem Bordcomputer |
| 30050 | Subroutine für die Ausgabe auf Radarschirm |
| 30060 | Subroutine die den Abstand zweier Punkte berechnet |
| 30070 | Subroutine die den Bordcomputermonitor löscht. |

**Listing zu »Unternehmen Galactica«. Die unterstrichenen Zeichen müssen mit »CTRL G« eingegeben werden.**

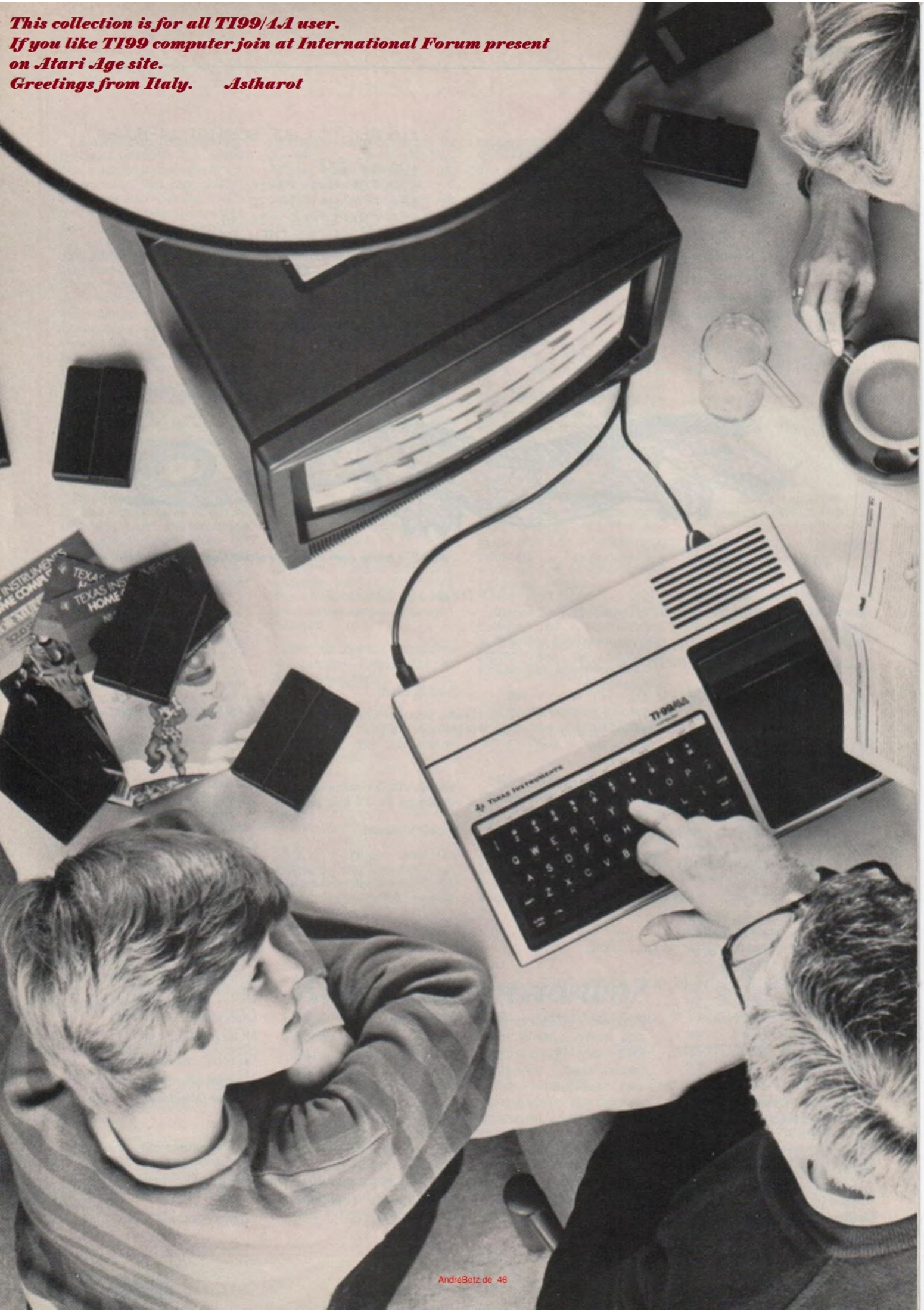
```
1000 RANDOMIZE :: OPTION BASE 1 :: ON ER
ROR 20001
1020 DIM A$(20),B$(8),C$(5),XZ(5),E(4),Y
Z(5),SX(8),SY(8),G$(8),K(5)
1030 WRZT$=CHR$(129)&CHR$(130)&CHR$(131)
&CHR$(132)&CHR$(133)&CHR$(134)&CHR$(140)
&CHR$(135)
1500 DATA 32,47,"FFC0A09088848383","8383
838383838383","83838488 90A0C0FF","FFFF00
00000000FF"
1510 DATA C1C12111090503FF,C1C1C1C1C1C1C
1C1,FF0305091121C1C1,FF0000000000FFFF,00
0406FFFF0604,002060FFFF602,1E0E1E3A70E04
1520 DATA 7870785C0E0702,18181818187E3C1
8,183C7E1818181818,02070E5C787078,40E070
3A1E0E1E
1530 DATA 123,142,"FFFFFFFF","8142240000
244281","0000000000001",,"00305F0E6C202C
",,"8"
1540 DATA 00000008,0004,1,00000240001,00
0000002000000 4,,024000303 0044,0008001818
122,004018180200008,,08114410A2144008,3C
7EFFFFFFFF7E3C,FFFFFF
1550 DATA 58,64
1552 DATA 00626408102646,,,,,,
1555 DATA 91,96
1560 DATA ,FFFFC3C3C3C3FFFF,0606030303 03
0606,183C7E7E3C18,0000003C,
1650 DATA 2,1,"time",7,1,"erde",2,18,"ga
l````:ok",4,18,"dollar",6,18,"jaeger",8,
18,"energie"
1700 DATA 1,2,5,10,1,8,10,10,1,19,12,10,
12,2,29,9
3000 RESTORE 1500 :: FOR U=1 TO 4 :: REA
D A,B :: FOR I=A TO B :: READ E$ :: CALL
 CHAR(I,E$):: NEXT I :: NEXT U
4000 CALL SCREEN(2)
4002 CALL COLOR(5,16,14,6,16,14,7,16,14,
8,16,14)
4003 CALL COLOR(9,13,4,10,13,4,11,13,4,1
2,13,4)
4005 CALL COLOR(3,2,12,4,2,12)
4010 CALL COLOR(0,2,2,1,16,6,2,16,4,13,1
6,5,14,15,5)
5000 CALL HCHAR(1,1,96,672):: CALL HCHAR
(22,1,64,96):: CALL VCHAR(1,32,31,48)
5010 RESTORE 1700 :: FOR I=1 TO 4 :: REA
D A,B,C,D :: CALL VCHAR(A,B,33,D):: CALL
 HCHAR(A+D,B,35,C):: CALL VCHAR(A,B+C,37
,D):: CALL HCHAR(A,B,39,C)
5020 CALL HCHAR(A,B,32):: CALL HCHAR(A+D
,B,34):: CALL HCHAR(A+D,B+C,36):: CALL H
CHAR(A,B+C,38):: NEXT I :: CALL D(-2,18,
"e\\\\\\\\\f")
5021 DISPLAY AT(23,1):"BITTE@HABEN@SIE@3
@MIN@GEDULD" :: RESTORE 5028 :: FOR Y=1
TO 8 :: READ B$(Y):: IF Y<6 THEN READ C$
(Y)
5022 NEXT Y :: READ D$ :: FOR Y=1 TO 20
:: A$(Y)="" :: FOR X=1 TO 30 :: IF RND<1
/(SQR(ABS((Y-10)^2)+ABS((X-15)^2))/1.6+1
)THEN A$(Y)=A$(Y)&SEG$(WRZT$,INT(RND*8)+
1,1)ELSE A$(Y)=A$(Y)&CHR$(135)
5024 NEXT X :: NEXT Y
5025 FOR I=1 TO 8 :: SX(I),A=INT(RND*30)
+1 :: SY(I),B=INT(RND*20)+1 :: A$(B)=SEG
$(A$(B),1,A-1)&CHR$(INT(RND*3)+136)&SEG$
(A$(B),A+1,30-A)
5026 NEXT I
5028 DATA orion,dracon,cygnus,lepton,cen
taurie,arkion,draconia,syrian,pegasus,my
onid,phoenix,altair,hydra,GGGGGGGGG,
5030 CALL SPRITE(#10,124,12,40,97,#26,12
3,12,171,152):: CALL MOTION(#26,0,30):: 
CALL HCHAR(22,1,30,32)
5050 RESTORE 1650 :: FOR I=1 TO 6 :: REA
D A,B,E$ :: CALL D(-12+A,B,E$):: NEXT I
6000 FOR I=1 TO 8 :: G$(I)="9"&STR$(INT(
RND*5+1))&STR$(INT(RND*9+1))&STR$(INT(RN
D*9+1)):: IF I<6 THEN XZ(I)=INT(RND*30)+
1 :: YZ(I)=INT(RND*30+1):: K(I)=10.9
6030 NEXT I :: P,FA,FB,FC,FD=0 :: L=15 :
: M=100 :: O=2000 :: N=1000
6040 XG=INT(RND*38-4):: XE=INT(RND*42-6)
:: IF XE-XG<16 THEN 6040 ELSE YG=INT(RND
*28-4)-3 :: YE=INT(RND*34-6)
6050 GOSUB 16000
7000 DISPLAY AT(23,1)BEEP:"1_SCHAEDEN@2_
ALARM@@3_SENSOR4_KURS@@5_VERHANDELN@@6_N
EXT"
8000 CALL I(123456,SP):: FOR U=1 TO 3 ::
 CALL SOUND(-65,U*250,6-U):: NEXT U :: C
ALL C
8010 DISPLAY AT(23,1):RPT$("@",39)&"9_ME
NUE@@@@@@@@@@" :: ON SP GOSUB 15000,1300
0,12000,11000,14000,16000
9000 CALL D(-10,23,"```"):: CALL D(-8,26,
"```"):: CALL D(-6,26,"```"):: IF FA=0 T
HEN FB,FC=0
9010 IF INT(N)<=0 OR INT(M)<=0 THEN CALL
 D(2,1,"galactica`zerstoert"):: GOTO 192
00 ELSE CALL L(4,2,P):: CALL L(2,22,M)::
 CALL L(4,25,O):: CALL L(6,25,L)
9500 CALL SPRITE(#22,93,2,73,155+N*.072)
:: GOTO 7000
11000 CALL D(1,9,"kurswechsel"):: CALL D
(3,8,"+`-`*```1`2`3"):: CALL D(5,8,")`0`
(```8`0`4"):: CALL D(7,8,".`,`/```7`6`5"
```

```
):: CALL I(1023456789,SP)
11021 KX=VAL(SEG$("-1000001010100-1-1",S
P*2-1,2)):: KY=VAL(SEG$("-100-1-10001010
100",SP*2-1,2)):: Q=VAL(SEG$("0384050200
07040601",SP*2-1,2))
11029 CALL PATTERN(#10,40+Q):: RETURN
12000 E$="`geortet" :: A=S+R :: IF A>0 T
HEN CALL D(1,1,STR$(A)&"`zylongeschwader
"&E$)ELSE IF S=0 THEN CALL D(1,1,"keine`
raumschiffe"&E$)
12020 IF T=0 THEN CALL D(3,3,"kein`sonne
nsystem"&E$):: RETURN
12030 IF FA=0 THEN CALL D(3,1,STR$(T)&"`
sonnensysteme"&E$):: RETURN
12040 CALL D(4,1,"planetenname`"&B$(FA))
:: IF SEG$(G$(FA),2,3)="000" THEN 12400
ELSE IF E(3)=1 THEN E$="kein" ELSE IF E(
3)<4 THEN E$="unter" ELSE IF E(3)<8 THEN
 E$="gleich" ELSE E$="hoeher"
12050 CALL D(6,1,E$&"`entwickeltes`leben
"):: IF E(3)>1 THEN CALL D(8,1,"nachkomm
en`der`"&C$(E(2))&"en")
12060 RETURN
12400 CALL D(7,1,"kultur`vernichtet"):: 
CALL D(8,1,"planet`verseucht"):: RETURN
13000 IF FD>0 THEN 14001 ELSE CALL D(1,1
,"schutzschrim```ein`1``aus`2"):: CALL I
(129,SP):: IF SP=1 THEN SI=M/10 ELSE IF
SP=2 THEN SI=0 ELSE RETURN
13010 CALL D(1,1,"angreifen``````ja``1`n
ein`2"):: CALL I(129,SP):: IF SP>1 THEN
RETURN
13020 FD=1 :: CALL D(1,1,"jaeger`einsetz
en`ja`1`nein`2"):: CALL I(129,SP):: IF S
P=2 THEN 13100 ELSE IF SP=9 THEN RETURN
13030 IF L=0 THEN CALL D(1,1,"alle`abfan
gjaeger`zerstoert`")ELSE GOSUB 13700
13040 KW=L*4 :: CALL D(1,1,"abfangjaeger
`gestartet"):: FOR I=1 TO 7 :: CALL SOUN
D(-75,-7,I*2):: NEXT I :: ON SP GOSUB 90
00,13800,13850,13900
13045 N=N-L*2 :: C=INT(RND*L*1/5)+1-INT(
RND+.5):: CALL D(0,1,STR$(C)&"`eigene`ja
eger`zerstoert"):: L=L-C
13100 CALL D(1,1,"laserkanonen``````ja`1`
nein`2"):: CALL I(129,SP):: IF SP>1 THEN
 RETURN ELSE GOSUB 13700
13110 KW=M/3 :: CALL D(1,1,"ziel`anvisie
rt"):: FOR U=1 TO 6 :: CALL SOUND(-100,U
*500,6-U):: NEXT U :: CALL D(2,1,"feuer`
frei"):: ON SP GOSUB 9000,13800,13850,13
900
13120 N=N-KW :: RETURN
13700 CALL C :: E$="9" :: CALL D(3,1,"zi
el`waehlen"):: IF FA>0 THEN IF SEG$(G$(F
A),2,3)<>"000" THEN CALL D(3,15,"1`plane
t"):: E$="1"
13705 IF R=0 THEN 13720 ELSE A=INT(K(RN)
):: IF K(RN)-A>0 THEN CALL D(4,15,"2`bas
isschiff"):: E$=E$&"2" ELSE E$=E$&"0"
13710 IF A>0 THEN CALL D(5,15,"3`zylonja
eger"):: E$=E$&"3"
13720 IF E$="9" THEN CALL D(3,1,"kein`zi
el`in`reichweite"):: GOTO 9000 ELSE CALL
 I(VAL("9"&E$),SP):: DISPLAY AT(13,1):RP
T$("`",140):: RETURN
13800 FB=1 :: B=E(1)*E(3):: GOSUB 13950
:: D=INT(E(1)-C/E(3))
13803 IF D>0 THEN G$(FA)=STR$(D)&SEG$(G$
(FA),2,3)ELSE G$(FA)=STR$(E(3))&"000" ::
 E(3)=0 :: CALL D(5,1,"planet`zerstoert"
):: RETURN
13805 IF D>6 OR RND>.6 OR E(3)<3 THEN RE
TURN ELSE CALL D(5,1,C$(E(2))&"en`wollen
`verhandeln")
13807 FB,FC=0 :: IF RND>.5 THEN E(4)=7 :
: G$(FA)=SEG$(G$(FA),1,3)&"7" ELSE E(4)=
0 :: G$(FA)=SEG$(G$(FA),1,3)&"0"
13810 RETURN
13850 B=15*(K(RN)-A):: GOSUB 13950 :: D=
INT(B-C)/10 :: IF D>0 AND D<1 THEN K(RN)
=A+D :: RETURN ELSE CALL D(5,1,"zylonsch
iff`vernichtet"):: R=0 :: I=RN :: GOSUB
18000
13875 IF A>0 THEN CALL D(7,1,STR$(A)&"`z
ylonjaeger`gefluechtet")
13878 RETURN
13900 B=A :: GOSUB 13950 :: D=INT(B-C)::
 IF D>0 THEN K(RN)=A-K(RN)+D :: RETURN E
LSE K(RN)=K(RN)-A :: CALL D(5,1,"zylonja
eger`vernichtet"):: RETURN
13950 GOSUB 18500 :: IF KW>2*B THEN E$="
schwerste" ELSE IF KW>B THEN E$="schwere
" ELSE IF KW<B/2 THEN E$="leichte" ELSE
E$=""
13955 CALL D(4,1,E$&"`treffer`beim`feind
"):: C=ABS(B-KW/2):: RETURN
14000 IF FC<>1 THEN 14003
14001 CALL D(1,1,"schon`durchgefuehrt"):
: RETURN
14003 FC=1 :: IF FA>0 THEN IF E(3)>1 THE
N 14009
14005 CALL D(1,1,"funkpeilung`negativ"):
: RETURN
14009 CALL D(1,1,"kontakt`mit`"&C$(E(2))
&"en`ist`ok")
14010 IF E(4)=0 THEN CALL D(3,1,C$(E(2))
&"en`ergeben`sich"):: C=1 :: GOTO 14015
14012 IF E(4)>2 THEN 14020 ELSE CALL D(3
,1,C$(E(2))&"en`sind`ausgestorben"):: IF
 RND>.5 THEN CALL D(4,1,"robotersysteme`
arbeiten`noch"):: C=1 ELSE 14016
14015 CALL D(6,1,"beiboote`landen`auf`pl
aneten"):: GOTO 14800
14016 CALL D(4,1,"enerrgiespeicher`noch`
ok"):: C=0 :: GOTO 14015
14020 IF FB>0 OR(E(4)>2 AND E(4)<7)THEN
FB=1 :: FC=0 :: IF RND>.7 THEN GOTO 1641
0 ELSE CALL D(8,1,"verlasse`orbit`sofort
"):: B=10 :: GOTO 16422
14030 CALL D(3,1,C$(E(2))&"en`helfen"):: 
 CALL D(8,1,"handeln``ja`1`nein`2"):: CA
LL I(129,SP):: IF SP>1 THEN FC=0 :: RETU
RN
14040 CALL C :: A=INT(E(1)*E(3)/5):: C=I
NT(A*(8+2*RND)):: B=INT(E(1)*E(3)*25):: 
D=1000-N :: IF B>D THEN B=INT(D)
14050 D=INT(D*(RND+.4)):: CALL D(1,1,"pr
eisangebot"):: CALL D(3,1,STR$(A)&"`jaeg
er`fuer`"&STR$(C)&"`dollar```"):: CALL D
(5,1,STR$(B)&":`energie`laden`"&STR$(D)&
"`dollar```")
14060 CALL D(8,1,"handeln`1``kaufen`2"):
: CALL I(129,SP):: IF SP=9 THEN RETURN E
```

```
LSE IF SP=1 THEN IF RND>.7 THEN FB=1 ::
GOTO 14020 ELSE 14040
14063 E$="jaeger" :: GOSUB 14075 :: IF S
P=1 THEN IF C>0 THEN CALL D(3,1,"geld`re
icht`nicht`aus``````")ELSE L=L+A :: 0=0-
C
14067 E$="energie" :: GOSUB 14075 :: IF
SP=1 THEN IF D>0 THEN CALL D(5,1,"geld`r
eicht`nicht`aus``````")ELSE N=N+B :: 0=
0-D
14070 GOTO 14850
14075 CALL D(8,1,E$&"`kaufen``ja`1``nein
`2"):: CALL I(129,SP):: IF SP=9 THEN 900
0 ELSE RETURN
14800 IF RND<.9 THEN A=INT(RND*(E(3)+E(1
))*C):: B=INT(E(1)*E(3)*15):: GOTO 14820
14805 GOSUB 18500 :: FB=1 :: FC=0 :: CAL
L D(7,1,"beiboote`der`galactica"):: CALL
 D(8,1,"von`kampfrobotern`vernichtet"):
 RETURN
14820 CALL D(7,1,STR$(A)&"`jaeger`und"):
: L=L+A :: C=1000-N :: IF B>C THEN B=C
14830 CALL D(7,15,STR$(INT(B/10))&"`:`en
ergie"):: B=INT(E(1)*E(3)*(5+RND*5)):: 0
=0+B :: CALL D(8,1,"und`"&STR$(B)&"`doll
ar`gefunden"):: N=N+C
14850 G$(FA)=STR$(INT((ASC(G$(FA))-48)/2
))&SEG$(G$(FA),2,3):: RETURN
15000 A=100-M :: IF A>.2*N THEN CALL D(3
,1,"energie`reicht`nicht`aus"):: RETURN
15020 CALL D(3,1,"schaeden`beseitigt"):
 N=N-A/.2 :: M=100 :: RETURN
16000 FD,FA,KW,T,R,S=0 :: P=P+1
16010 XG=XG+KX :: YG=YG+KY :: FOR I=YG-4
 TO YG+4 :: IF I>20 OR I<1 OR XG<-3 OR X
G>34 THEN E$=D$ :: GOTO 16090
16020 IF XG>4.9 AND XG<27 THEN E$=SEG$(A
$(I),XG-4,9)ELSE IF XG<5 THEN E$=SEG$(D$
,1,5-XG)&SEG$(A$(I),1,4+XG)ELSE E$=SEG$(
A$(I),XG-4,35-XG)&D$
16090 DISPLAY AT(I-YG+6,7)SIZE(9):E$ ::
NEXT I :: CALL DELSPRITE(#16,#17)
16100 CALL A(XG,YG,XE,YE,A):: DISPLAY AT
(9,3)SIZE(2):"``" :: CALL L(9,2,INT(A)):
: IF A<4 THEN C=(XE-XG+12)*8 :: D=(YE-YG
+5)*8 :: CALL SPRITE(#17,141,16,D,C,#16,
127,4,D,C)
16105 IF A<1 AND KX=0 AND KY=0 THEN 1900
0 ELSE CALL GCHAR(6,13,A):: IF A=140 THE
N 19100
16200 FOR I=1 TO 8 :: CALL A(XG,YG,SX(I)
,SY(I),A):: IF A<5 THEN T=T+1 :: IF A<1
AND KX=0 AND KY=0 THEN FA=I :: CALL D(1,
1,"galactica`im`orbit")
16202 IF I>5 THEN 16405 ELSE IF YZ(I)>YG
 THEN A=-1 ELSE A=1
16203 IF XZ(I)>XG THEN B=-1 ELSE B=1
16205 XZ(I)=XZ(I)+B :: YZ(I)=YZ(I)+A ::
K(I)=K(I)+1 :: CALL A(XG,YG,XZ(I),YZ(I),
A):: IF A>4.9 THEN 16220 ELSE B=YZ(I)-YG
+6
16210 C=XZ(I)-XG+13 :: RN=I :: CALL SPRI
TE(#10+I,94,12,B*8-6,C*8-7):: CALL GCHAR
(B,C,D):: IF D=140 THEN 16900 ELSE R=R+1
 :: D=INT(K(I)):: KW=KW+2*D+30*(K(I)-D)/
(A+1):: GOSUB 18500
16215 GOTO 16405
16220 IF A<8 THEN S=S+1 :: CALL D(4,1,"a
nfliegende`objekte`geortet")
16405 NEXT I :: IF R>0 THEN CALL D(5,1,"
angriff`durch`zylonen")
16407 IF FA=0 THEN 16415
16410 FOR I=1 TO 4 :: E(I)=VAL(SEG$(G$(F
A),I,1)):: NEXT I :: IF (E(4)<7 OR FB>0)
AND E(3)>1 THEN CALL D(1,1,"angriff`vom`
planeten"):: KW=KW+E(1)*E(3):: GOSUB 185
00
16415 IF KW<=0 THEN 16420 ELSE B=10 :: I
F SI=0 THEN E$="aktivieren" ELSE IF KW<2
*SI THEN E$="standgehalten" :: KW=0 ELSE
 E$="durchbrochen" :: KW=KW-2*SI
16418 D=INT(KW/5):: M=M-D :: CALL D(7,1,
"schutzschirm`"&E$):: CALL D(8,1,STR$(D)
&"`schaeden`durch`treffer"):: GOTO 1642
2
16420 IF S=0 THEN B=12 ELSE B=4
16422 N=N-(ABS(KX)+ABS(KY))*7-SI*2 :: CA
LL COLOR(#26,B):: GOTO 9000
16900 E$="zylonen``" :: D=2 :: GOSUB 170
00
16910 GOSUB 18000
16920 GOTO 16220
17000 CALL D(D,1,E$&"`im`asteroidenfeld"
):: CALL D(D+1,1,"zerschellt"):: RETURN
18000 FOR U=1 TO 3 :: CALL SOUND(75,-7,U
*2):: CALL COLOR(#10+RN,U+10):: CALL PAT
TERN(#RN+10,143-U*3):: NEXT U :: CALL DE
LSPRITE(#RN+10)
18010 D=SGN(RND-.5):: XZ(RN)=XZ(RN)+D*IN
T(RND*8+6):: YZ(RN)=YZ(RN)+D*INT(RND*6+7
):: K(RN)=0.9 :: RETURN
18500 FOR U=1 TO 4 :: CALL COLOR(13,16,1
3-U*2,14,8,13-U*2):: CALL SOUND(-125,110
,30,110,30,20000/U,30,-8,U):: NEXT U ::
RETURN
19000 CALL D(2,1,"ziel`erreicht"):: GOTO
 19200
19100 E$="galactica" :: D=2 :: GOSUB 170
00
19200 DISPLAY AT(16,1):RPT$("`",140):: C
ALL DELSPRITE(ALL):: CALL D(5,8,"neues`s
piel``1"):: CALL D(7,8,"neue``karte``2")
:: CALL I(12,SP):: ON SP GOTO 5030,5021
20001 ON ERROR 20001 :: GOTO 9000
30000 SUB I(A,SP):: A$=STR$(A)
30001 CALL SOUND(-4250,-7,25):: FOR I=1
TO 25 :: CALL KEY(0,K,S):: IF S<>0 THEN
SP=POS(A$,CHR$(K),1):: IF SP>0 THEN 3001
0
30003 NEXT I :: CALL SOUND(-50,4000,0)::
 GOTO 30001
30010 CALL SOUND(-50,770,3):: SUBEND
30020 SUB L(A,B,C):: CALL D(-12+A,B,STR$
(C)):: SUBEND
30050 SUB D(Y,X,A$):: DISPLAY AT(Y+12,X)
SIZE(LEN(A$)):A$ :: SUBEND
30060 SUB A(A,B,C,D,E):: E=SQR(ABS(A-C)^
2+ABS(B-D)^2):: SUBEND
30070 SUB C :: DISPLAY AT(13,1):RPT$("`"
,224):: SUBEND
```

**Listing zu »Unternehmen Galactica« (Schluß).**
**Die unterstrichenen Zeichen müssen mit »CTRL G« eingegeben werden.**

***This collection is for all TI99/4A user.***
***If you like TI99 computer join at International Forum present on Atari Age site.***
***Greetings from Italy.  Astharot***




AndreBetz.de 46